EMNLP 2009

Proceedings of the 2009 Conference on

Empirical Methods in Natural Language Processing

A meeting of SIGDAT,
a special interest group of ACL
held in conjunction with ACL-IJCNLP 2009

6–7 August 2009
Singapore

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

The conference organizers are grateful to Microsoft Research for their generous support.

Microsoft®
Research

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-932432-59-6 / 1-932432-59-0 (Volume 1)
ISBN 978-1-932432-62-6 / 1-932432-62-0 (Volume 2)
ISBN 978-1-932432-63-3 / 1-932432-63-9 (Volume 3)

Introduction

Welcome to the 2009 Conference on Empirical Methods in Natural Language Processing!

The conference is organized under the auspices of SIGDAT, the ACL Special Interest Group for linguistic data and corpus-based approaches to natural language processing. It is co-located this year with ACL-IJCNLP 2009 in Singapore.

EMNLP received 475 submissions, a new record. We were able to accept 163 papers in total (an acceptance rate of 34%). Of these, 96 (20%) were accepted for oral presentation, and 67 (14%) for poster presentation. The papers were selected by a program committee of 15 area chairs, from Asia, Europe, and North America, assisted by a panel of 389 reviewers. This year EMNLP again held an author response period. Authors were able to read and respond to the reviews of their paper before the program committee made a final decision. They were asked to correct factual errors in the reviews and answer questions raised in the reviewer comments.The intention was to help produce more accurate reviews. In some cases, reviewers changed their scores in view of the authors response and the area chairs read all responses carefully prior to making recommendations for acceptance.

First and foremost, we would like to thank the authors who submitted their work to EMNLP. The sheer number of submissions reflects how broad and active our field is. We are deeply indebted to the area chairs and the reviewers for their hard work. They enabled us to select an exciting program and to provide valuable feedback to the authors. Additional thanks to the Publications Chair, David Chiang, who put this volume together. Jason Eisner helped us immensely by compiling a web site on "How to Serve as Program Chair of a Conference."[1] Special thanks to David Yarowsky and Ken Church of SIGDAT who provided much valuable advice and assistance over the past months. We are akso grateful for the financial support from Microsoft.

We are most grateful to Haizhou Li who helped us with various logistic and organizational aspects of the conference. Rich Gerber and the START team responded to our questions quickly, and helped us manage the large number of submissions smoothly. Finally, thanks are due to our predecessors, Mirella Lapata and Hwee Tou Ng, whose experience and example we shamelessly exploited.

Philipp Koehn
Rada Mihalcea

[1] http://www.cs.jhu.edu/jason/advice/how-to-chair-a-conference.html

Program Co-Chairs:

Philipp Koehn, University of Edinburgh
Rada Mihalcea, University of North Texas

Area Chairs:

Stephen Clark, University of Cambridge
Mona Diab, Columbia University
Jason Eisner, Johns Hopkins University
Katrin Erk, University of Texas
Eric Fosler-Lussier, Ohio State University
Iryna Gurevych, Darmstadt University
Hang Li, Microsoft
Chin-Yew Lin, Microsoft
Adam Lopez, University of Edinburgh
Vivi Năstase, EML Research
Miles Osborne, University of Edinburgh
Tim Paek, Microsoft
Marius Paşca, Google
Carlo Strapparava, FBK-irst
Theresa Wilson, University of Edinburgh

Local Arrangements Chair:

Haizhou Li, Institute for Infocomm Research

Publications Chair:

David Chiang, USC Information Sciences Institute

Reviewers:

Ahmed Abbasi
Eugene Agichtein
Amr Ahmed
Ben Allison
Yasemin Altun
Alina Andreevskaia
Shlomo Argamon
Abhishek Arun
Jordi Atserias
Necip Fazil Ayan
Timothy Baldwim
Srinivas Bangalore
Roy Bar-Haim
Regina Barzilay
Roberto Basili
Sabine Bergler
Steven Bethard
Rahul Bhagat
Chris Biemann
Alexandra Birch
Phil Blunsom
Dan Bohus
Gemma Boleda
Johan Bos
Alexandre Bouchard-Côté
Thorsten Brants
Eric Breck
Sam Brody
Susan Brown
Paul Buitelaar
Razvan Bunescu
Aljoscha Burchardt
Jill Burstein
Bill Byrne
Chris Callison-Burch
Nicola Cancedda
Yunbo Cao
Guiseppe Carenini
Marine Carpuat
Xavier Carreras
John Carroll
Vitor Carvalho
Francisco Casacuberta
Mauro Cettolo
Nate Chambers
Yee Seng Chan
Pi-Chuan Chang
Harr Chen
Hsin-Hsi Chen
Pu-Jen Cheng
Xueqi Cheng
Colin Cherry
David Chiang
Yejin Choi
Jennifer Chu-Carroll
Ken Church
Massimiliano Ciaramita
James Clarke
Paul Cook
Bonaventura Coppola
Marta R. Costa-Jussa
Mathias Creutz
Montse Cuadros Oller
Silviu Cucerzan
Hang Cui
James Curran
Walter Daelemans
Hercules Dalianis
Dipanjan Das
Dmitry Davidov
Adrià de Gispert
Marie-Catherine de Marneffe
Steve DeNeefe
John DeNero
Yonggang Deng
Pascal Denis
Ann Devitt
Fernando Diaz
Markus Dickinson
Mark Dredze
Markus Dreyer
Amit Dubey
Chris Dyer
Koji Eguchi
Andreas Eisele
Michael Elhadad
Andrea Esuli
Stefan Evert
Richard Farkas
Afsaneh Fazly
Christiane Fellbaum
Jenny Finkel
Margaret Fleck
Radu Florian
George Foster
Anette Frank
Alex Fraser
Pascale Fung
Evgeniy Gabrilovich
Michel Galley
Michael Gamon
Jianfeng Gao
Yuqing Gao
Nikesh Garera
Kallirroi Georgila
Ulrich Germann
Daniel Gildea
Jesús Giménez
Jonathan Ginzburg
Roxana Girju
Claudio Giuliano
Jim Glass
Alfio Gliozzo
Andrew Goldberg
John Goldsmith
Sharon Goldwater
Carlos Gomez-Rodriguez
Julio Gonzalo
Cyril Goutte
Mark Greenwood
Nizar Habash
Barry Haddow
Patrick Haffner
Thomas Hain
Dilek Hakkani-Tur
David Hall
Keith Hall
Sanda Harabagiu
Saša Hasan
Mark Hasegawa-Johnson
Xiaodong He
Jeffrey Heinz
James Henderson
John Henderson
Graeme Hirst
Hieu Hoang
Julia Hockenmaier
Kristy Hollingshead
Tracy Holloway King
Florentina Hristea
Liang Huang
Lluis Hurtado
Nancy Ide
Gonzalo Iglesias
Diana Inkpen
Martin Jansche
Nitin Jindal
Richard Johansson
Pamela Jordan
Joemon Jose
Hiroshi Kanayama
Rohit Kate
Graham Katz
Tatsuya Kawahara
Frank Keller
Adam Kilgarriff
Soo-Min Kim
Dietrich Klakow
Alexandre Klementiev
Kevin Knight
Rob Koeling
Grzegorz Kondrak
Terry Koo
Moshe Koppel
András Kornai
Zornitsa Kozareva
Lun-Wei Ku
Sandra Kuebler
Roland Kuhn
Ravi Kumar
Shankar Kumar
Sadao Kurohashi
Wei Lai
Wai Lam
Lori Lamel
Alon Lavie
Victor Lavrenko
Matthew Lease
Gary Lee
Gina-Anne Levow
Wei Li
Zhifei Li
Dekang Lin
Hsuan-Tien Lin
Jimmy Lin
Shou-de Lin
Lucian Lita
Bing Liu
Nathan Liu
Qun Liu
Yang Liu

Karen Livescu
Wolfgang Macherey
Nitin Madnani
Rob Malouf
Gideon Mann
Chris Manning
Daniel Marcu
Katja Markert
David Martinez
Andre Martins
Yuji Matsumoto
Takuya Matsuzaki
Mark Maybury
Andrew McCallum
Diana McCarthy
Ryan McDonald
Kathy McKeown
Qiaozhu Mei
Arul Menezes
Florian Metze
Donald Metzler
Haitao Mi
Einat Minkov
Shachar Mirkin
Jeff Mitchell
Vibhu Mittal
Yusuke Miyao
Marie-Francine Moens
Saif Mohammad
Christof Monz
Tsun Moon
Raymond Mooney
Robert Moore
Alessandro Moschitti
Tony Mullen
Smaranda Muresan
Gabriel Murray
Gabriele Musillo
Sung-Hyon Myaeng
Mikio Nakano
Preslav Nakov
Su Nam Kim
Alexis Nasr
Tetsuya Nasukawa
Roberto Navigli
Mark-Jan Nederhof
Ani Nenkova
Gunther Neumann
Hwee Tou Ng
Vincent Ng
Nicolas Nicolov
Rodney Nielsen
Joakim Nivre
Tadashi Nomoto
Franz Och
Kemal Oflazer
Manubu Okumura
Jahna Otterbacher
Iadh Ounis
Ulrike Pado
Lluis Padro
Bo Pang
Patrick Pantel
Fuchun Peng
Marco Pennacchiotti
Slav Petrov
Emanuele Pianta
Olivier Pietquin
Daniele Pighin
Massimo Poesio
Elias Ponvert
Simone Ponzetto
Ana-Maria Popescu
Marius Popescu
Maja Popović
Victor Poznanski
John Prager
Rashmi Prasad
Partha Pratim Talukdar
Matthew Purver
Silvia Quarteroni
Chris Quirk
Stephan Raaijmakers
Dan Ramage
Owen Rambow
Delip Rao
Ari Rappoport
Deepak Ravichandran
Roi Reichart
Sebastian Riedel
Stefan Riezler
German Rigau
Michael Riley
Hae-Chang Rim
Laura Rimell
Brian Roark
Paolo Rosso
Antti-Veikko Rosti
Alex Rudnicky
Anna Rumshisky
Vasile Rus
Anton Rytting
Marta Sabou
Bogdan Sacaleanu
Kenji Sagae
Horacio Saggion
Murat Saraclar
Anoop Sarkar
Roser Sauri
Helmut Schmid
Karin Schuler
Sabine Schulte im Walde
Holger Schwenk
Fabrizio Sebastiani
Frederique Segond
Yohei Seki
Satoshi Sekine
Stephanie Seneff
Hendra Setiawan
Burr Settles
Izhak Shafran
Dou Shen
Khalil Sima'an
Michel Simard
David Smith
Noah Smith
Matthew Snover
Swapna Somasundaran
Young-In Song
Aitor Soroa
Caroline Sporleder
Richard Sproat
Rohini Srihari
Mark Steedman
Amanda Stent
Mark Stevenson
Veselin Stoyanov
Michael Strube
Eiichiro Sumita
Maosong Sun
Mihai Surdeanu
Gyrgy Szarvas
Stan Szpakowicz
Idan Szpektor
Hiroya Takamura
David Talbot
Joel Tetreault
Simone Teufel
Christoph Tillmann
Ivan Titov
Kristina Toutanova
Roy Tromble
Yoshimasa Tsuruoka
Antal van den Bosch
Gertjan van Noord
Josef van Genabith
Lonneke van der Plas
Benjamin Van Durme
Lucy Vanderwende
Sebastian Varges
Ashish Venugopal
David Vilar
Stephan Vogel
Piek Vossen
Kuansan Wang
Richard Wang
Wei Wang
Taro Watanabe
Andy Way
Amy Weinberg
David Weir
Dan Weld
Michael White
Richard Wicentowski
Jan Wiebe
Jason Williams
Shuly Wintner
Rene Witte
Wensi Xi
Fei Xia
Deyi Xiong
Gu Xu
Jun Xu
Deniz Yuret
Richard Zens
Dell Zhang
Hao Zhang
Min Zhang
Yue Zhang
Bing Zhao
Min Zhao
Tie-Jun Zhao
Ming Zhou
Shenghuo Zhu
Imed Zitouni
Onno Zoeter
Andreas Zollmann
Ingrid Zukerman

Table of Contents

Conference Program

Thursday, August 06, 2009

8:45–9:00 Opening remarks

9:00–10:00 Invited Talk

10:00–10:30 Coffee Break

Session 1A (Theatre): Semantic Parsing

10:30–10:55 *Unsupervised Semantic Parsing*
Hoifung Poon and Pedro Domingos

10:55–11:20 *Graph Alignment for Semi-Supervised Semantic Role Labeling*
Hagen Fürstenau and Mirella Lapata

11:20–11:45 *Semi-supervised Semantic Role Labeling Using the Latent Words Language Model*
Koen Deschacht and Marie-Francine Moens

11:45–12:10 *Semantic Dependency Parsing of NomBank and PropBank: An Efficient Integrated Approach via a Large-scale Feature Selection*
Hai Zhao, Wenliang Chen and Chunyu Kit

Session 1B (MR208): Machine Translation I

10:30–10:55 *First- and Second-Order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests*
Zhifei Li and Jason Eisner

10:55–11:20 *Feasibility of Human-in-the-loop Minimum Error Rate Training*
Omar F. Zaidan and Chris Callison-Burch

11:20–11:45 *Cube Pruning as Heuristic Search*
Mark Hopkins and Greg Langmead

11:45–12:10 *Effective Use of Linguistic and Contextual Information for Statistical Machine Translation*
Libin Shen, Jinxi Xu, Bing Zhang, Spyros Matsoukas and Ralph Weischedel

Thursday, August 06, 2009 (continued)

Session 1C (MR209): Machine Learning and Statistical Models I

10:30–10:55 *Active Learning by Labeling Features*
Gregory Druck, Burr Settles and Andrew McCallum

10:55–11:20 *Efficient Kernels for Sentence Pair Classification*
Fabio Massimo Zanzotto and Lorenzo Dell'Arciprete

11:20–11:45 *Graphical Models over Multiple Strings*
Markus Dreyer and Jason Eisner

11:45–12:10 *Reverse Engineering of Tree Kernel Feature Spaces*
Daniele Pighin and Alessandro Moschitti

Session 1D (MR203): Information Extraction

10:30–10:55 *A Rich Feature Vector for Protein-Protein Interaction Extraction from Multiple Corpora*
Makoto Miwa, Rune Sætre, Yusuke Miyao and Jun'ichi Tsujii

10:55–11:20 *Generalized Expectation Criteria for Bootstrapping Extractors using Record-Text Alignment*
Kedar Bellare and Andrew McCallum

11:20–11:45 *Nested Named Entity Recognition*
Jenny Rose Finkel and Christopher D. Manning

11:45–12:10 *A Unified Model of Phrasal and Sentential Evidence for Information Extraction*
Siddharth Patwardhan and Ellen Riloff

12:10–13:50 Lunch

Thursday, August 06, 2009 (continued)

Session 2A (Theatre): Subjectivity and Sentiment I

13:50–14:15 *Review Sentiment Scoring via a Parse-and-Paraphrase Paradigm*
Jingjing Liu and Stephanie Seneff

14:15–14:40 *Supervised and Unsupervised Methods in Employing Discourse Relations for Improving Opinion Polarity Classification*
Swapna Somasundaran, Galileo Namata, Janyce Wiebe and Lise Getoor

14:40–15:05 *Sentiment Analysis of Conditional Sentences*
Ramanathan Narayanan, Bing Liu and Alok Choudhary

15:05–15:30 *Subjectivity Word Sense Disambiguation*
Cem Akkaya, Janyce Wiebe and Rada Mihalcea

Session 2B (MR208): Machine Translation II

13:50–14:15 *Non-Projective Parsing for Statistical Machine Translation*
Xavier Carreras and Michael Collins

14:15–14:40 *Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models*
Arne Mauser, Saša Hasan and Hermann Ney

14:40–15:05 *Feature-Rich Translation by Quasi-Synchronous Lattice Parsing*
Kevin Gimpel and Noah A. Smith

15:05–15:30 *Improved Word Alignment with Statistics and Linguistic Heuristics*
Ulf Hermjakob

Thursday, August 06, 2009 (continued)

Session 2C (MR209): Natural Language Processing for Web 2.0

13:50–14:15 *Entity Extraction via Ensemble Semantics*
Marco Pennacchiotti and Patrick Pantel

14:15–14:40 *Labeled LDA: A Supervised Topic Model for Credit Attribution in Multi-labeled Corpora*
Daniel Ramage, David Hall, Ramesh Nallapati and Christopher D. Manning

14:40–15:05 *Clustering to Find Exemplar Terms for Keyphrase Extraction*
Zhiyuan Liu, Peng Li, Yabin Zheng and Maosong Sun

15:05–15:30 *Geo-mining: Discovery of Road and Transport Networks Using Directional Patterns*
Dmitry Davidov and Ari Rappoport

Session 2D (MR203): Language Resources and Evaluation

13:50–14:15 *Wikipedia as Frame Information Repository*
Sara Tonelli and Claudio Giuliano

14:15–14:40 *Fast, Cheap, and Creative: Evaluating Translation Quality Using Amazon's Mechanical Turk*
Chris Callison-Burch

14:40–15:05 *How Well Does Active Learning* Actually *Work? Time-based Evaluation of Cost-reduction Strategies for Language Documentation*
Jason Baldridge and Alexis Palmer

15:05–15:30 *Automatically Evaluating Content Selection in Summarization without Human Models*
Annie Louis and Ani Nenkova

15:30–16:00 Coffee Break

Thursday, August 06, 2009 (continued)

Session 3A (Theatre): Discourse and Dialogue

16:00–16:25 *Classifier Combination for Contextual Idiom Detection Without Labelled Data*
Linlin Li and Caroline Sporleder

16:25–16:50 *Deriving Lexical and Syntactic Expectation-based Measures for Psycholinguistic Modeling via Incremental Top-down Parsing*
Brian Roark, Asaf Bachrach, Carlos Cardenas and Christophe Pallier

16:50–17:15 *It's Not You, it's Me: Detecting Flirting and its Misperception in Speed-Dates*
Rajesh Ranganath, Dan Jurafsky and Dan McFarland

17:15–17:40 *Recognizing Implicit Discourse Relations in the Penn Discourse Treebank*
Ziheng Lin, Min-Yen Kan and Hwee Tou Ng

Session 3B (MR208): Machine Translation III

16:00–16:25 *A Bayesian Model of Syntax-Directed Tree to String Grammar Induction*
Trevor Cohn and Phil Blunsom

16:25–16:50 *Better Synchronous Binarization for Machine Translation*
Tong Xiao, Mu Li, Dongdong Zhang, Jingbo Zhu and Ming Zhou

16:50–17:15 *Accuracy-Based Scoring for DOT: Towards Direct Error Minimization for Data-Oriented Translation*
Daniel Galron, Sergio Penkale, Andy Way and I. Dan Melamed

17:15–17:40 *Improved Statistical Machine Translation Using Monolingually-Derived Paraphrases*
Yuval Marton, Chris Callison-Burch and Philip Resnik

Thursday, August 06, 2009 (continued)

Session 3C (MR209): Summarization and Generation

16:00–16:25 *A Comparison of Model Free versus Model Intensive Approaches to Sentence Compression*
Tadashi Nomoto

16:25–16:50 *Natural Language Generation with Tree Conditional Random Fields*
Wei Lu, Hwee Tou Ng and Wee Sun Lee

16:50–17:15 *Perceptron Reranking for CCG Realization*
Michael White and Rajakrishnan Rajkumar

17:15–17:40 *Multi-Document Summarisation Using Generic Relation Extraction*
Ben Hachey

Session 3D (MR203): Lexical Semantics I

16:00–16:25 *Language Models Based on Semantic Composition*
Jeff Mitchell and Mirella Lapata

16:25–16:50 *Graded Word Sense Assignment*
Katrin Erk and Diana McCarthy

16:50–17:15 *Joint Learning of Preposition Senses and Semantic Roles of Prepositional Phrases*
Daniel Dahlmeier, Hwee Tou Ng and Tanja Schultz

17:15–17:40 *Projecting Parameters for Multilingual Word Sense Disambiguation*
Mitesh M. Khapra, Sapan Shah, Piyush Kedia and Pushpak Bhattacharyya

Thursday, August 06, 2009 (continued)

18:00–20:00 **Poster Session and Reception**

Gazpacho and Summer Rash: Lexical Relationships from Temporal Patterns of Web Search Queries
Enrique Alfonseca, Massimiliano Ciaramita and Keith Hall

A Compact Forest for Scalable Inference over Entailment and Paraphrase Rules
Roy Bar-Haim, Jonathan Berant and Ido Dagan

Discriminative Substring Decoding for Transliteration
Colin Cherry and Hisami Suzuki

Re-Ranking Models Based-on Small Training Data for Spoken Language Understanding
Marco Dinarelli, Alessandro Moschitti and Giuseppe Riccardi

Empirical Exploitation of Click Data for Task Specific Ranking
Anlei Dong, Yi Chang, Shihao Ji, Ciya Liao, Xin Li and Zhaohui Zheng

The Feature Subspace Method for SMT System Combination
Nan Duan, Mu Li, Tong Xiao and Ming Zhou

Lattice-based System Combination for Statistical Machine Translation
Yang Feng, Yang Liu, Haitao Mi, Qun Liu and Yajuan Lü

A Joint Language Model With Fine-grain Syntactic Tags
Denis Filimonov and Mary Harper

Bidirectional Phrase-based Statistical Machine Translation
Andrew Finch and Eiichiro Sumita

Real-time Decision Detection in Multi-party Dialogue
Matthew Frampton, Jia Huang, Trung Bui and Stanley Peters

On the Role of Lexical Features in Sequence Labeling
Yoav Goldberg and Michael Elhadad

Thursday, August 06, 2009 (continued)

18:00–20:00 **Poster Session and Reception (continued)**

Simple Coreference Resolution with Rich Syntactic and Semantic Features
Aria Haghighi and Dan Klein

Descriptive and Empirical Approaches to Capturing Underlying Dependencies among Parsing Errors
Tadayoshi Hara, Yusuke Miyao and Jun'ichi Tsujii

Large-Scale Verb Entailment Acquisition from the Web
Chikara Hashimoto, Kentaro Torisawa, Kow Kuroda, Stijn De Saeger, Masaki Murata and Jun'ichi Kazama

A Syntactified Direct Translation Model with Linear-time Decoding
Hany Hassan, Khalil Sima'an and Andy Way

Cross-lingual Semantic Relatedness Using Encyclopedic Knowledge
Samer Hassan and Rada Mihalcea

Joint Optimization for Machine Translation System Combination
Xiaodong He and Kristina Toutanova

Fully Lexicalising CCGbank with Hat Categories
Matthew Honnibal and James R. Curran

Bilingually-Constrained (Monolingual) Shift-Reduce Parsing
Liang Huang, Wenbin Jiang and Qun Liu

Accurate Semantic Class Classifier for Coreference Resolution
Zhiheng Huang, Guangping Zeng, Weiqun Xu and Asli Celikyilmaz

Real-Word Spelling Correction using Google Web 1T 3-grams
Aminul Islam and Diana Inkpen

Semi-supervised Speech Act Recognition in Emails and Forums
Minwoo Jeong, Chin-Yew Lin and Gary Geunbae Lee

Thursday, August 06, 2009 (continued)

18:00–20:00 **Poster Session and Reception (continued)**

Using Morphological and Syntactic Structures for Chinese Opinion Analysis
Lun-Wei Ku, Ting-Hao Huang and Hsin-Hsi Chen

Finding Short Definitions of Terms on Web Pages
Gerasimos Lampouras and Ion Androutsopoulos

Improving Nominal SRL in Chinese Language with Verbal SRL Information and Automatic Predicate Recognition
Junhui Li, Guodong Zhou, Hai Zhao, Qiaoming Zhu and Peide Qian

On the Use of Virtual Evidence in Conditional Random Fields
Xiao Li

Refining Grammars for Parsing with Hierarchical Semantic Knowledge
Xiaojun Lin, Yang Fan, Meng Zhang, Xihong Wu and Huisheng Chi

Bayesian Learning of Phrasal Tree-to-String Templates
Ding Liu and Daniel Gildea

Human-competitive Tagging Using Automatic Keyphrase Extraction
Olena Medelyan, Eibe Frank and Ian H. Witten

Supervised Learning of a Probabilistic Lexicon of Verb Semantic Classes
Yusuke Miyao and Jun'ichi Tsujii

A Study on the Semantic Relatedness of Query and Document Terms in Information Retrieval
Christof Müller and Iryna Gurevych

Predicting Subjectivity in Multimodal Conversations
Gabriel Murray and Giuseppe Carenini

Improved Statistical Machine Translation for Resource-Poor Languages Using Related Resource-Rich Languages
Preslav Nakov and Hwee Tou Ng

Thursday, August 06, 2009 (continued)

18:00–20:00 **Poster Session and Reception (continued)**

What's in a Name? In Some Languages, Grammatical Gender
Vivi Nastase and Marius Popescu

Convolution Kernels on Constituent, Dependency and Sequential Structures for Relation Extraction
Truc-Vien T. Nguyen, Alessandro Moschitti and Giuseppe Riccardi

Automatic Acquisition of the Argument-Predicate *Relations from a Frame-Annotated Corpus*
Ekaterina Ovchinnikova, Theodore Alexandrov and Tonio Wandmacher

Detecting Speculations and their Scopes in Scientific Text
Arzucan Özgür and Dragomir R. Radev

Cross-Cultural Analysis of Blogs and Forums with Mixed-Collection Topic Models
Michael Paul and Roxana Girju

Consensus Training for Consensus Decoding in Machine Translation
Adam Pauls, John Denero and Dan Klein

Using Word-Sense Disambiguation Methods to Classify Web Queries by Intent
Emily Pitler and Ken Church

Semi-Supervised Learning for Semantic Relation Classification using Stratified Sampling Strategy
Longhua Qian, Guodong Zhou, Fang Kong and Qiaoming Zhu

Construction of a Blog Emotion Corpus for Chinese Emotional Expression Analysis
Changqin Quan and Fuji Ren

A Probabilistic Model for Associative Anaphora Resolution
Ryohei Sasano and Sadao Kurohashi

Quantifier Scope Disambiguation Using Extracted Pragmatic Knowledge: Preliminary Results
Prakash Srinivasan and Alexander Yates

Thursday, August 06, 2009 (continued)

18:00–20:00 **Poster Session and Reception (continued)**

Chinese Semantic Role Labeling with Shallow Parsing
Weiwei Sun, Zhifang Sui, Meng Wang and Xin Wang

Discovery of Term Variation in Japanese Web Search Queries
Hisami Suzuki, Xiao Li and Jianfeng Gao

Towards Domain-Independent Argumentative Zoning: Evidence from Chemistry and Computational Linguistics
Simone Teufel, Advaith Siddharthan and Colin Batchelor

Character-level Analysis of Semi-Structured Documents for Set Expansion
Richard C. Wang and William W. Cohen

Classifying Relations for Biomedical Named Entity Disambiguation
Xinglong Wang, Jun'ichi Tsujii and Sophia Ananiadou

Domain adaptive bootstrapping for named entity recognition
Dan Wu, Wee Sun Lee, Nan Ye and Hai Leong Chieu

Phrase Dependency Parsing for Opinion Mining
Yuanbin Wu, Qi Zhang, Xuangjing Huang and Lide Wu

Polynomial to Linear: Efficient Classification with Conjunctive Features
Naoki Yoshinaga and Masaru Kitsuregawa

K-Best Combination of Syntactic Parsers
Hui Zhang, Min Zhang, Chew Lim Tan and Haizhou Li

Chinese Novelty Mining
Yi Zhang and Flora S. Tsai

Latent Document Re-Ranking
Dong Zhou and Vincent Wade

Friday, August 07, 2009

Session 4A (Theatre): Multi-word Expressions

8:45–9:10 *Multi-Word Expression Identification Using Sentence Surface Features*
Ram Boukobza and Ari Rappoport

9:10–9:35 *Acquiring Translation Equivalences of Multiword Expressions by Normalized Correlation Frequencies*
Ming-Hong Bai, Jia-Ming You, Keh-Jiann Chen and Jason S. Chang

9:35–10:00 *Collocation Extraction Using Monolingual Word Alignment Method*
Zhanyi Liu, Haifeng Wang, Hua Wu and Sheng Li

Session 4B (MR208): Machine Learning and Statistical Models II

8:45–9:10 *Multi-Class Confidence Weighted Algorithms*
Koby Crammer, Mark Dredze and Alex Kulesza

9:10–9:35 *Model Adaptation via Model Interpolation and Boosting for Web Search Ranking*
Jianfeng Gao, Qiang Wu, Chris Burges, Krysta Svore, Yi Su, Nazan Khan, Shalin Shah and Hongyan Zhou

9:35–10:00 *A Structural Support Vector Method for Extracting Contexts and Answers of Questions from Online Forums*
Wen-Yun Yang, Yunbo Cao and Chin-Yew Lin

Session 4C (MR209): Information Retrieval and Questions Answering

8:45–9:10 *Mining Search Engine Clickthrough Log for Matching N-gram Features*
Huihsin Tseng, Longbin Chen, Fan Li, Ziming Zhuang, Lei Duan and Belle Tseng

9:10–9:35 *The Role of Named Entities in Web People Search*
Javier Artiles, Enrique Amigó and Julio Gonzalo

9:35–10:00 *Investigation of Question Classifier in Question Answering*
Zhiheng Huang, Marcus Thint and Asli Celikyilmaz

Friday, August 07, 2009 (continued)

Session 4D (MR203): Syntax and Parsing I

8:45–9:10 *An Empirical Study of Semi-supervised Structured Conditional Models for Dependency Parsing*
Jun Suzuki, Hideki Isozaki, Xavier Carreras and Michael Collins

9:10–9:35 *Statistical Bistratal Dependency Parsing*
Richard Johansson

9:35–10:00 *Improving Dependency Parsing with Subtrees from Auto-Parsed Data*
Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto and Kentaro Torisawa

10:00–10:30 Coffee Break

Session 5A (Theatre): Subjectivity and Sentiment II

10:30–10:55 *Topic-wise, Sentiment-wise, or Otherwise? Identifying the Hidden Dimension for Unsupervised Text Classification*
Sajib Dasgupta and Vincent Ng

10:55–11:20 *Adapting a Polarity Lexicon using Integer Linear Programming for Domain-Specific Sentiment Classification*
Yejin Choi and Claire Cardie

11:20–11:45 *Generating High-Coverage Semantic Orientation Lexicons From Overtly Marked Words and a Thesaurus*
Saif Mohammad, Cody Dunne and Bonnie Dorr

11:45–12:10 *Matching Reviews to Objects using a Language Model*
Nilesh Dalvi, Ravi Kumar, Bo Pang and Andrew Tomkins

Friday, August 07, 2009 (continued)

Session 5B (MR208): Lexical Semantics II

10:30–10:55 *EEG Responds to Conceptual Stimuli and Corpus Semantics*
Brian Murphy, Marco Baroni and Massimo Poesio

10:55–11:20 *A Comparison of Windowless and Window-Based Computational Association Measures as Predictors of Syntagmatic Human Associations*
Justin Washtell and Katja Markert

11:20–11:45 *Improving Verb Clustering with Automatically Acquired Selectional Preferences*
Lin Sun and Anna Korhonen

11:45–12:10 *Improving Web Search Relevance with Semantic Features*
Yumao Lu, Fuchun Peng, Gilad Mishne, Xing Wei and Benoit Dumoulin

Session 5C (MR209): Phonology and Morphology

10:30–10:55 *Can Chinese Phonemes Improve Machine Transliteration?: A Comparative Study of English-to-Chinese Transliteration Models*
Jong-Hoon Oh, Kiyotaka Uchimoto and Kentaro Torisawa

10:55–11:20 *Unsupervised Morphological Segmentation and Clustering with Document Boundaries*
Taesun Moon, Katrin Erk and Jason Baldridge

11:20–11:45 *The Infinite HMM for Unsupervised PoS Tagging*
Jurgen Van Gael, Andreas Vlachos and Zoubin Ghahramani

11:45–12:10 *A Simple Unsupervised Learner for POS Disambiguation Rules Given Only a Minimal Lexicon*
Qiuye Zhao and Mitch Marcus

Friday, August 07, 2009 (continued)

Session 5D (MR203): Machine Translation IV

10:30–10:55 *Tree Kernel-based SVM with Structured Syntactic Knowledge for BTG-based Phrase Reordering*
Min Zhang and Haizhou Li

10:55–11:20 *Discriminative Corpus Weight Estimation for Machine Translation*
Spyros Matsoukas, Antti-Veikko I. Rosti and Bing Zhang

11:20–11:45 *Unsupervised Tokenization for Machine Translation*
Tagyoung Chung and Daniel Gildea

11:45–12:10 *Synchronous Tree Adjoining Machine Translation*
Steve DeNeefe and Kevin Knight

12:10–13:50 Lunch

Session 6A (Theatre): Speech and Language Modeling

13:50–14:15 *Word Buffering Models for Improved Speech Repair Parsing*
Tim Miller

14:15–14:40 *Less is More: Significance-Based N-gram Selection for Smaller, Better Language Models*
Robert C. Moore and Chris Quirk

14:40–15:05 *Stream-based Randomised Language Models for SMT*
Abby Levenberg and Miles Osborne

15:05–15:30 *Integrating Sentence- and Word-level Error Identification for Disfluency Correction*
Erin Fitzgerald, Frederick Jelinek and Keith Hall

Friday, August 07, 2009 (continued)

Session 6B (MR208): Semantic Similarity

13:50–14:15 *Estimating Semantic Distance Using Soft Semantic Constraints in Knowledge-Source – Corpus Hybrid Models*
Yuval Marton, Saif Mohammad and Philip Resnik

14:15–14:40 *Recognizing Textual Relatedness with Predicate-Argument Structures*
Rui Wang and Yi Zhang

14:40–15:05 *Learning Term-weighting Functions for Similarity Measures*
Wen-tau Yih

15:05–15:30 *A Relational Model of Semantic Similarity between Words using Automatically Extracted Lexical Pattern Clusters from the Web*
Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka

Session 6C (MR209): Syntax and Parsing II

13:50–14:15 *Unbounded Dependency Recovery for Parser Evaluation*
Laura Rimell, Stephen Clark and Mark Steedman

14:15–14:40 *Parser Adaptation and Projection with Quasi-Synchronous Grammar Features*
David A. Smith and Jason Eisner

14:40–15:05 *Self-Training PCFG Grammars with Latent Annotations Across Languages*
Zhongqiang Huang and Mary Harper

15:05–15:30 *An Alternative to Head-Driven Approaches for Parsing a (Relatively) Free Word-Order Language*
Reut Tsarfaty, Khalil Sima'an and Remko Scha

Friday, August 07, 2009 (continued)

Session 6D (MR203): Multilinguality

13:50–14:15 *Enhancement of Lexical Concepts Using Cross-lingual Web Mining*
Dmitry Davidov and Ari Rappoport

14:15–14:40 *Bilingual Dictionary Generation for Low-resourced Language Pairs*
István Varga and Shoichi Yokoyama

14:40–15:05 *Multilingual Spectral Clustering Using Document Similarity Propagation*
Dani Yogatama and Kumiko Tanaka-Ishii

15:05–15:30 *Polylingual Topic Models*
David Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith and Andrew McCallum

15:30–16:00 Coffee Break

Session 7A (Theatre): Natural Language Applications

16:00–16:25 *Using the Web for Language Independent Spellchecking and Autocorrection*
Casey Whitelaw, Ben Hutchinson, Grace Y Chung and Ged Ellis

16:25–16:50 *Statistical Estimation of Word Acquisition with Application to Readability Prediction*
Paul Kidwell, Guy Lebanon and Kevyn Collins-Thompson

16:50–17:15 *Combining Collocations, Lexical and Encyclopedic Knowledge for Metonymy Resolution*
Vivi Nastase and Michael Strube

17:15–17:40 *Segmenting Email Message Text into Zones*
Andrew Lampert, Robert Dale and Cécile Paris

Friday, August 07, 2009 (continued)

Session 7B (MR208): Lexical Semantics III

16:00–16:25 *Hypernym Discovery Based on Distributional Similarity and Hierarchical Structures*
Ichiro Yamada, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, Masaki Murata, Stijn De Saeger, Francis Bond and Asuka Sumida

16:25–16:50 *Web-Scale Distributional Similarity and Entity Set Expansion*
Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu and Vishnu Vyas

16:50–17:15 *Toward Completeness in Concept Extraction and Classification*
Eduard Hovy, Zornitsa Kozareva and Ellen Riloff

17:15–17:40 *Reading to Learn: Constructing Features from Semantic Abstracts*
Jacob Eisenstein, James Clarke, Dan Goldwasser and Dan Roth

Session 7C (MR209): Coreference Resolution

16:00–16:25 *Supervised Models for Coreference Resolution*
Altaf Rahman and Vincent Ng

16:25–16:50 *Global Learning of Noun Phrase Anaphoricity in Coreference Resolution via Label Propagation*
GuoDong Zhou and Fang Kong

16:50–17:15 *Employing the Centering Theory in Pronoun Resolution from the Semantic Perspective*
Fang Kong, GuoDong Zhou and Qiaoming Zhu

17:15–17:40 *Person Cross Document Coreference with Name Perplexity Estimates*
Octavian Popescu

Friday, August 07, 2009 (continued)

Session 7D (MR203): Machine Translation V

16:00–16:25 *Learning Linear Ordering Problems for Better Translation*
Roy Tromble and Jason Eisner

16:25–16:50 *Weighted Alignment Matrices for Statistical Machine Translation*
Yang Liu, Tian Xia, Xinyan Xiao and Qun Liu

16:50–17:15 *Sinuhe – Statistical Machine Translation using a Globally Trained Conditional Exponential Family Translation Model*
Matti Kääriäinen

17:15–17:40 *Fast Translation Rule Matching for Syntax-based Statistical Machine Translation*
Hui Zhang, Min Zhang, Haizhou Li and Chew Lim Tan

Unsupervised Semantic Parsing

Hoifung Poon **Pedro Domingos**
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350, U.S.A.
{hoifung,pedrod}@cs.washington.edu

Abstract

We present the first unsupervised approach to the problem of learning a semantic parser, using Markov logic. Our USP system transforms dependency trees into quasi-logical forms, recursively induces lambda forms from these, and clusters them to abstract away syntactic variations of the same meaning. The MAP semantic parse of a sentence is obtained by recursively assigning its parts to lambda-form clusters and composing them. We evaluate our approach by using it to extract a knowledge base from biomedical abstracts and answer questions. USP substantially outperforms TextRunner, DIRT and an informed baseline on both precision and recall on this task.

1 Introduction

Semantic parsing maps text to formal meaning representations. This contrasts with semantic role labeling (Carreras and Marquez, 2004) and other forms of shallow semantic processing, which do not aim to produce complete formal meanings. Traditionally, semantic parsers were constructed manually, but this is too costly and brittle. Recently, a number of machine learning approaches have been proposed (Zettlemoyer and Collins, 2005; Mooney, 2007). However, they are supervised, and providing the target logical form for each sentence is costly and difficult to do consistently and with high quality. Unsupervised approaches have been applied to shallow semantic tasks (e.g., paraphrasing (Lin and Pantel, 2001), information extraction (Banko et al., 2007)), but not to semantic parsing.

In this paper we develop the first unsupervised approach to semantic parsing, using Markov logic (Richardson and Domingos, 2006). Our USP system starts by clustering tokens of the same type, and then recursively clusters expressions whose subexpressions belong to the same clusters. Experiments on a biomedical corpus show that this approach is able to successfully translate syntactic variations into a logical representation of their common meaning (e.g., USP learns to map active and passive voice to the same logical form, etc.). This in turn allows it to correctly answer many more questions than systems based on TextRunner (Banko et al., 2007) and DIRT (Lin and Pantel, 2001).

We begin by reviewing the necessary background on semantic parsing and Markov logic. We then describe our Markov logic network for unsupervised semantic parsing, and the learning and inference algorithms we used. Finally, we present our experiments and results.

2 Background

2.1 Semantic Parsing

The standard language for formal meaning representation is first-order logic. A term is any expression representing an object in the domain. An atomic formula or atom is a predicate symbol applied to a tuple of terms. Formulas are recursively constructed from atomic formulas using logical connectives and quantifiers. A *lexical entry* defines the logical form for a lexical item (e.g., a word). The semantic parse of a sentence is derived by starting with logical forms in the lexical entries and recursively composing the meaning of larger fragments from their parts. In traditional approaches, the lexical entries and meaning-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1–10,
Singapore, 6-7 August 2009.

composition rules are both manually constructed. Below are sample rules in a definite clause grammar (DCG) for parsing the sentence: "Utah borders Idaho".

$$Verb[\lambda \mathtt{y} \lambda \mathtt{x}.\mathtt{borders}(\mathtt{x},\mathtt{y})] \rightarrow borders$$
$$NP[\mathtt{Utah}] \rightarrow Utah$$
$$NP[\mathtt{Idaho}] \rightarrow Idaho$$
$$VP[\mathtt{rel}(\mathtt{obj})] \rightarrow Verb[\mathtt{rel}] \quad NP[\mathtt{obj}]$$
$$S[\mathtt{rel}(\mathtt{obj})] \rightarrow NP[\mathtt{obj}] \quad VP[\mathtt{rel}]$$

The first three lines are lexical entries. They are fired upon seeing the individual words. For example, the first rule applies to the word "borders" and generates syntactic category *Verb* with the meaning $\lambda \mathtt{y} \lambda \mathtt{x}.\mathtt{borders}(\mathtt{x},\mathtt{y})$ that represents the next-to relation. Here, we use the standard lambda-calculus notation, where $\lambda \mathtt{y} \lambda \mathtt{x}.\mathtt{borders}(\mathtt{x},\mathtt{y})$ represents a function that is true for any $(\mathtt{x},\mathtt{y})$-pair such that $\mathtt{borders}(\mathtt{x},\mathtt{y})$ holds. The last two rules compose the meanings of sub-parts into that of the larger part. For example, after the first and third rules are fired, the fourth rule fires and generates $VP[\lambda \mathtt{y} \lambda \mathtt{x}.\mathtt{borders}(\mathtt{x},\mathtt{y})(\mathtt{Idaho})]$; this meaning simplifies to $\lambda \mathtt{x}.\mathtt{borders}(\mathtt{x},\mathtt{Idaho})$ by the *λ-reduction rule*, which substitutes the argument for a variable in a functional application.

A major challenge to semantic parsing is syntactic variations of the same meaning, which abound in natural languages. For example, the aforementioned sentence can be rephrased as "Utah is next to Idaho,""Utah shares a border with Idaho," etc. Manually encoding all these variations into the grammar is tedious and error-prone. Supervised semantic parsing addresses this issue by learning to construct the grammar automatically from sample meaning annotations (Mooney, 2007). Existing approaches differ in the meaning representation languages they use and the amount of annotation required. In the approach of Zettlemoyer and Collins (2005), the training data consists of sentences paired with their meanings in lambda form. A probabilistic combinatory categorial grammar (PCCG) is learned using a log-linear model, where the probability of the final logical form L and meaning-derivation tree T conditioned on the sentence S is $P(L,T|S) = \frac{1}{Z}\exp\left(\sum_i w_i f_i(L,T,S)\right)$. Here Z is the normalization constant and f_i are the feature functions with weights w_i. Candidate lexical entries are generated by a domain-specific procedure based on the target logical forms.

The major limitation of supervised approaches is that they require meaning annotations for example sentences. Even in a restricted domain, doing this consistently and with high quality requires nontrivial effort. For unrestricted text, the complexity and subjectivity of annotation render it essentially infeasible; even pre-specifying the target predicates and objects is very difficult. Therefore, to apply semantic parsing beyond limited domains, it is crucial to develop unsupervised methods that do not rely on labeled meanings.

In the past, unsupervised approaches have been applied to some semantic tasks, but not to semantic parsing. For example, DIRT (Lin and Pantel, 2001) learns paraphrases of binary relations based on distributional similarity of their arguments; TextRunner (Banko et al., 2007) automatically extracts relational triples in open domains using a self-trained extractor; SNE applies relational clustering to generate a semantic network from TextRunner triples (Kok and Domingos, 2008). While these systems illustrate the promise of unsupervised methods, the semantic content they extract is nonetheless shallow and does not constitute the complete formal meaning that can be obtained by a semantic parser.

Another issue is that existing approaches to semantic parsing learn to parse syntax and semantics together.[1] The drawback is that the complexity in syntactic processing is coupled with semantic parsing and makes the latter even harder. For example, when applying their approach to a different domain with somewhat less rigid syntax, Zettlemoyer and Collins (2007) need to introduce new combinators and new forms of candidate lexical entries. Ideally, we should leverage the enormous progress made in syntactic parsing and generate semantic parses directly from syntactic analysis.

2.2 Markov Logic

In many NLP applications, there exist rich relations among objects, and recent work in statistical relational learning (Getoor and Taskar, 2007) and structured prediction (Bakir et al., 2007) has shown that leveraging these can greatly improve accuracy. One of the most powerful representations for this is Markov logic, which is a probabilistic extension of first-order logic (Richardson and Domingos, 2006). Markov logic makes it

[1] The only exception that we are aware of is Ge and Mooney (2009).

possible to compactly specify probability distributions over complex relational domains, and has been successfully applied to unsupervised coreference resolution (Poon and Domingos, 2008) and other tasks. A *Markov logic network (MLN)* is a set of weighted first-order clauses. Together with a set of constants, it defines a Markov network with one node per ground atom and one feature per ground clause. The weight of a feature is the weight of the first-order clause that originated it. The probability of a state x in such a network is given by the log-linear model $P(x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right)$, where Z is a normalization constant, w_i is the weight of the ith formula, and n_i is the number of satisfied groundings.

3 Unsupervised Semantic Parsing with Markov Logic

Unsupervised semantic parsing (**USP**) rests on three key ideas. First, the target predicate and object constants, which are pre-specified in supervised semantic parsing, can be viewed as clusters of syntactic variations of the same meaning, and can be learned from data. For example, `borders` represents the next-to relation, and can be viewed as the cluster of different forms for expressing this relation, such as "borders", "is next to", "share the border with"; `Utah` represents the state of Utah, and can be viewed as the cluster of "Utah", "the beehive state", etc.

Second, the identification and clustering of candidate forms are integrated with the learning for meaning composition, where forms that are used in composition with the same forms are encouraged to cluster together, and so are forms that are composed of the same sub-forms. This amounts to a novel form of relational clustering, where clustering is done not just on fixed elements in relational tuples, but on arbitrary forms that are built up recursively.

Third, while most existing approaches (manual or supervised learning) learn to parse both syntax and semantics, unsupervised semantic parsing starts directly from syntactic analyses and focuses solely on translating them to semantic content. This enables us to leverage advanced syntactic parsers and (indirectly) the available rich resources for them. More importantly, it separates the complexity in syntactic analysis from the semantic one, and makes the latter much easier to perform. In particular, meaning composition does not require domain-specific procedures for generating candidate lexicons, as is often needed by supervised methods.

The input to our USP system consists of dependency trees of training sentences. Compared to phrase-structure syntax, dependency trees are the more appropriate starting point for semantic processing, as they already exhibit much of the relation-argument structure at the lexical level.

USP first uses a deterministic procedure to convert dependency trees into quasi-logical forms (QLFs). The QLFs and their sub-formulas have natural lambda forms, as will be described later. Starting with clusters of lambda forms at the atom level, USP recursively builds up clusters of larger lambda forms. The final output is a probability distribution over lambda-form clusters and their compositions, as well as the MAP semantic parses of training sentences.

In the remainder of the section, we describe the details of USP. We first present the procedure for generating QLFs from dependency trees. We then introduce their lambda forms and clusters, and show how semantic parsing works in this setting. Finally, we present the Markov logic network (MLN) used by USP. In the next sections, we present efficient algorithms for learning and inference with this MLN.

3.1 Derivation of Quasi-Logical Forms

A *dependency tree* is a tree where nodes are words and edges are dependency labels. To derive the QLF, we convert each node to an unary atom with the predicate being the lemma plus POS tag (below, we still use the word for simplicity), and each edge to a binary atom with the predicate being the dependency label. For example, the node for Utah becomes $\texttt{Utah}(\texttt{n}_1)$ and the subject dependency becomes $\texttt{nsubj}(\texttt{n1}, \texttt{n2})$. Here, the $\texttt{n}_\texttt{i}$ are Skolem constants indexed by the nodes. The QLF for a sentence is the conjunction of the atoms for the nodes and edges, e.g., the sentence above will become $\texttt{borders}(\texttt{n}_1) \wedge \texttt{Utah}(\texttt{n}_2) \wedge \texttt{Idaho}(\texttt{n}_3) \wedge \texttt{nsubj}(\texttt{n}_1, \texttt{n}_2) \wedge \texttt{dobj}(\texttt{n}_1, \texttt{n}_3)$.

3.2 Lambda-Form Clusters and Semantic Parsing in USP

Given a QLF, a relation or an object is represented by the conjunction of a subset of the atoms. For example, the next-to relation is represented by $\texttt{borders}(\texttt{n}_1) \wedge \texttt{nsubj}(\texttt{n}_1, \texttt{n}_2) \wedge \texttt{dobj}(\texttt{n}_1, \texttt{n}_3)$, and the states of Utah and Idaho are represented

by $\texttt{Utah}(\texttt{n}_2)$ and $\texttt{Idaho}(\texttt{n}_3)$. The meaning composition of two sub-formulas is simply their conjunction. This allows the maximum flexibility in learning. In particular, lexical entries are no longer limited to be adjacent words as in Zettlemoyer and Collins (2005), but can be arbitrary fragments in a dependency tree.

For every sub-formula F, we define a corresponding lambda form that can be derived by replacing every Skolem constant $\texttt{n}_\texttt{i}$ that does not appear in any unary atom in F with a unique lambda variable $\texttt{x}_\texttt{i}$. Intuitively, such constants represent objects introduced somewhere else (by the unary atoms containing them), and correspond to the arguments of the relation represented by F. For example, the lambda form for $\texttt{borders}(\texttt{n}_1) \wedge \texttt{nsubj}(\texttt{n}_1, \texttt{n}_2) \wedge \texttt{dobj}(\texttt{n}_1, \texttt{n}_3)$ is $\lambda\texttt{x}_2\lambda\texttt{x}_3.\ \texttt{borders}(\texttt{n}_1) \wedge \texttt{nsubj}(\texttt{n}_1, \texttt{x}_2) \wedge \texttt{dobj}(\texttt{n}_1, \texttt{x}_3)$.

Conceptually, a lambda-form cluster is a set of semantically interchangeable lambda forms. For example, to express the meaning that Utah borders Idaho, we can use any form in the cluster representing the next-to relation (e.g., "borders", "shares a border with"), any form in the cluster representing the state of Utah (e.g., "the beehive state"), and any form in the cluster representing the state of Idaho (e.g., "Idaho"). Conditioned on the clusters, the choices of individual lambda forms are independent of each other.

To handle variable number of arguments, we follow Davidsonian semantics and further decompose a lambda form into the *core form*, which does not contain any lambda variable (e.g., $\texttt{borders}(\texttt{n}_1)$), and the *argument forms*, which contain a single lambda variable (e.g., $\lambda\texttt{x}_2.\texttt{nsubj}(\texttt{n}_1, \texttt{x}_2)$ and $\lambda\texttt{x}_3.\texttt{dobj}(\texttt{n}_1, \texttt{x}_3)$). Each lambda-form cluster may contain some number of *argument types*, which cluster distinct forms of the same argument in a relation. For example, in Stanford dependencies, the object of a verb uses the dependency `dobj` in the active voice, but `nsubjpass` in passive.

Lambda-form clusters abstract away syntactic variations of the same meaning. Given an instance of cluster $\texttt{T}$ with arguments of argument types $\texttt{A}_1, \cdots, \texttt{A}_\texttt{k}$, its *abstract lambda form* is given by $\lambda\texttt{x}_1 \cdots \lambda\texttt{x}_\texttt{k}.\texttt{T}(\texttt{n}) \wedge \bigwedge_{\texttt{i}=1}^{\texttt{k}} \texttt{A}_\texttt{i}(\texttt{n}, \texttt{x}_\texttt{i})$.

Given a sentence and its QLF, semantic parsing amounts to partitioning the atoms in the QLF, dividing each part into core form and argument forms, and then assigning each form to a cluster or an argument type. The final logical form is derived by composing the abstract lambda forms of the parts using the λ-reduction rule.[2]

3.3 The USP MLN

Formally, for a QLF Q, a semantic parse L partitions Q into parts $\texttt{p}_1, \texttt{p}_2, \cdots, \texttt{p}_\texttt{n}$; each part $\texttt{p}$ is assigned to some lambda-form cluster $\texttt{c}$, and is further partitioned into core form $\texttt{f}$ and argument forms $\texttt{f}_1, \cdots, \texttt{f}_\texttt{k}$; each argument form is assigned to an argument type $\texttt{a}$ in $\texttt{c}$. The USP MLN defines a joint probability distribution over Q and L by modeling the distributions over forms and arguments given the cluster or argument type.

Before presenting the predicates and formulas in our MLN, we should emphasize that they should not be confused with the atoms and formulas in the QLFs, which are represented by reified constants and variables.

To model distributions over lambda forms, we introduce the predicates $\texttt{Form}(\texttt{p}, \texttt{f}!)$ and $\texttt{ArgForm}(\texttt{p}, \texttt{i}, \texttt{f}!)$, where $\texttt{p}$ is a part, $\texttt{i}$ is the index of an argument, and $\texttt{f}$ is a QLF subformula. $\texttt{Form}(\texttt{p}, \texttt{f})$ is true iff part $\texttt{p}$ has core form $\texttt{f}$, and $\texttt{ArgForm}(\texttt{p}, \texttt{i}, \texttt{f})$ is true iff the $\texttt{i}$th argument in $\texttt{p}$ has form $\texttt{f}$.[3] The "$\texttt{f}!$" notation signifies that each part or argument can have only one form.

To model distributions over arguments, we introduce three more predicates: $\texttt{ArgType}(\texttt{p}, \texttt{i}, \texttt{a}!)$ signifies that the $\texttt{i}$th argument of $\texttt{p}$ is assigned to argument type $\texttt{a}$; $\texttt{Arg}(\texttt{p}, \texttt{i}, \texttt{p}')$ signifies that the $\texttt{i}$th argument of $\texttt{p}$ is $\texttt{p}'$; $\texttt{Number}(\texttt{p}, \texttt{a}, \texttt{n})$ signifies that there are $\texttt{n}$ arguments of $\texttt{p}$ that are assigned to type $\texttt{a}$. The truth value of $\texttt{Number}(\texttt{p}, \texttt{a}, \texttt{n})$ is determined by the `ArgType` atoms.

Unsupervised semantic parsing can be captured by four formulas:

$$\texttt{p} \in +\texttt{c} \wedge \texttt{Form}(\texttt{p}, +\texttt{f})$$
$$\texttt{ArgType}(\texttt{p}, \texttt{i}, +\texttt{a}) \wedge \texttt{ArgForm}(\texttt{p}, \texttt{i}, +\texttt{f})$$
$$\texttt{Arg}(\texttt{p}, \texttt{i}, \texttt{p}') \wedge \texttt{ArgType}(\texttt{p}, \texttt{i}, +\texttt{a}) \wedge \texttt{p}' \in +\texttt{c}'$$
$$\texttt{Number}(\texttt{p}, +\texttt{a}, +\texttt{n})$$

All free variables are implicitly universally quantified. The "+" notation signifies that the MLN contains an instance of the formula, with a separate weight, for each value combination of the

[2] Currently, we do not handle quantifier scoping or semantics for specific closed-class words such as determiners. These will be pursued in future work.

[3] There are hard constraints to guarantee that these assignments form a legal partition. We omit them for simplicity.

variables with a plus sign. The first formula models the mixture of core forms given the cluster, and the others model the mixtures of argument forms, argument types, and argument numbers, respectively, given the argument type.

To encourage clustering and avoid overfitting, we impose an exponential prior with weight α on the number of parameters.[4]

The MLN above has one problem: it often clusters expressions that are semantically opposite. For example, it clusters antonyms like "elderly/young", "mature/immature". This issue also occurs in other semantic-processing systems (e.g., DIRT). In general, this is a difficult open problem that only recently has started to receive some attention (Mohammad et al., 2008). Resolving this is not the focus of this paper, but we describe a general heuristic for fixing this problem. We observe that the problem stems from the lack of negative features for discovering meanings in contrast. In natural languages, parallel structures like conjunctions are one such feature.[5] We thus introduce an exponential prior with weight β on the number of conjunctions where the two conjunctive parts are assigned to the same cluster. To detect conjunction, we simply used the Stanford dependencies that begin with "conj". This proves very effective, fixing the majority of the errors in our experiments.

4 Inference

Given a sentence and the quasi-logical form Q derived from its dependency tree, the conditional probability for a semantic parse L is given by $Pr(L|Q) \propto \exp\left(\sum_i w_i n_i(L,Q)\right)$. The MAP semantic parse is simply $\arg\max_L \sum_i w_i n_i(L,Q)$. Enumerating all L's is intractable. It is also unnecessary, since most partitions will result in parts whose lambda forms have no cluster they can be assigned to. Instead, USP uses a greedy algorithm to search for the MAP parse. First we introduce some definitions: a partition is called *λ-reducible from* $\mathtt{p}$ if it can be obtained from the current partition by recursively λ-reducing the part containing $\mathtt{p}$ with one of its arguments; such a partition is called *feasible* if the core form of the new part is contained in some cluster. For example, consider the QLF of "Utah borders Idaho" and assume that the current partition is $\lambda \mathtt{x}_2\mathtt{x}_3.\mathtt{borders}(\mathtt{n}_1) \wedge \mathtt{nsubj}(\mathtt{n}_1,\mathtt{x}_2) \wedge \mathtt{dobj}(\mathtt{n}_1,\mathtt{x}_3)$, $\mathtt{Utah}(\mathtt{n}_2)$, $\mathtt{Idaho}(\mathtt{n}_3)$. Then the following partition is λ-reducible from the first part in the above partition: $\lambda \mathtt{x}_3.\mathtt{borders}(\mathtt{n}_1) \wedge \mathtt{nsubj}(\mathtt{n}_1,\mathtt{n}_2) \wedge \mathtt{Utah}(\mathtt{n}_2) \wedge \mathtt{dobj}(\mathtt{n}_1,\mathtt{x}_3)$, $\mathtt{Idaho}(\mathtt{n}_3)$. Whether this new partition is feasible depends on whether the core form of the new part $\lambda \mathtt{x}_3.\mathtt{borders}(\mathtt{n}_1) \wedge \mathtt{nsubj}(\mathtt{n}_1,\mathtt{n}_2) \wedge \mathtt{Utah}(\mathtt{n}_2) \wedge \mathtt{dobj}(\mathtt{n}_1,\mathtt{x}_3)$ (i.e. $\mathtt{borders}(\mathtt{n}_1) \wedge \mathtt{nsubj}(\mathtt{n}_1,\mathtt{n}_2) \wedge \mathtt{Utah}(\mathtt{n}_2)$) is contained in some lambda-form cluster.

Algorithm 1 USP-Parse(*MLN, QLF*)

```
Form parts for individual atoms in QLF and assign each to its most probable cluster
repeat
  for all parts p in the current partition do
    for all partitions that are λ-reducible from p and feasible do
      Find the most probable cluster and argument type assignments for the new part and its arguments
    end for
  end for
  Change to the new partition and assignments with the highest gain in probability
until none of these improve the probability
return current partition and assignments
```

Algorithm 1 gives pseudo-code for our algorithm. Given part $\mathtt{p}$, finding partitions that are λ-reducible from $\mathtt{p}$ and feasible can be done in time $O(ST)$, where S is the size of the clustering in the number of core forms and T is the maximum number of atoms in a core form. We omit the proof here but point out that it is related to the unordered subtree matching problem which can be solved in linear time (Kilpelainen, 1992). Inverted indexes (e.g., from $\mathtt{p}$ to eligible core forms) are used to further improve the efficiency. For a new part $\mathtt{p}$ and a cluster that contains $\mathtt{p}$'s core form, there are k^m ways of assigning $\mathtt{p}$'s m arguments to the k argument types of the cluster. For larger k and m, this is very expensive. We therefore approximate it by assigning each argument to the best type, independent of other arguments.

This algorithm is very efficient, and is used repeatedly in learning.

[4] Excluding weights of ∞ or $-\infty$, which signify hard constraints.

[5] For example, in the sentence "IL-2 inhibits X in A and induces Y in B", the conjunction between "inhibits" and "induces" suggests that they are different. If "inhibits" and "induces" are indeed synonyms, such a sentence will sound awkward and would probably be rephrased as "IL-2 inhibits X in A and Y in B".

5 Learning

The learning problem in USP is to maximize the log-likelihood of observing the QLFs obtained from the dependency trees, denoted by Q, summing out the unobserved semantic parses:

$$\begin{aligned} L_\theta(Q) &= \log P_\theta(Q) \\ &= \log \textstyle\sum_L P_\theta(Q, L) \end{aligned}$$

Here, L are the semantic parses, θ are the MLN parameters, and $P_\theta(Q, L)$ are the completion likelihoods. A serious challenge in unsupervised learning is the identifiability problem (i.e., the optimal parameters are not unique) (Liang and Klein, 2008). This problem is particularly severe for log-linear models with hard constraints, which are common in MLNs. For example, in our USP MLN, conditioned on the fact that $\mathtt{p} \in \mathtt{c}$, there is exactly one value of $\mathtt{f}$ that can satisfy the formula $\mathtt{p} \in \mathtt{c} \wedge \mathtt{Form}(\mathtt{p}, \mathtt{f})$, and if we add some constant number to the weights of $\mathtt{p} \in \mathtt{c} \wedge \mathtt{Form}(\mathtt{p}, \mathtt{f})$ for all $\mathtt{f}$, the probability distribution stays the same.[6] The learner can be easily confused by the infinitely many optima, especially in the early stages. To address this problem, we impose local normalization constraints on specific groups of formulas that are mutually exclusive and exhaustive, i.e., in each group, we require that $\sum_{i=1}^{k} e^{w_i} = 1$, where w_i are the weights of formulas in the group. Grouping is done in such a way as to encourage the intended mixture behaviors. Specifically, for the rule $\mathtt{p} \in +\mathtt{c} \wedge \mathtt{Form}(\mathtt{p}, +\mathtt{f})$, all instances given a fixed $\mathtt{c}$ form a group; for each of the remaining three rules, all instances given a fixed $\mathtt{a}$ form a group. Notice that with these constraints the completion likelihood $P(Q, L)$ can be computed in closed form for any L. In particular, each formula group contributes a term equal to the weight of the currently satisfied formula. In addition, the optimal weights that maximize the completion likelihood $P(Q, L)$ can be derived in closed form using empirical relative frequencies. E.g., the optimal weight of $\mathtt{p} \in \mathtt{c} \wedge \mathtt{Form}(\mathtt{p}, \mathtt{f})$ is $\log(n_{\mathtt{c},\mathtt{f}}/n_{\mathtt{c}})$, where $n_{\mathtt{c},\mathtt{f}}$ is the number of parts $\mathtt{p}$ that satisfy both $\mathtt{p} \in \mathtt{c}$ and $\mathtt{Form}(\mathtt{p}, \mathtt{f})$, and $n_{\mathtt{c}}$ is the number of parts $\mathtt{p}$ that satisfy $\mathtt{p} \in \mathtt{c}$.[7] We leverage this fact for efficient learning in USP.

[6] Regularizations, e.g., Gaussian priors on weights, alleviate this problem by penalizing large weights, but it remains true that weights within a short range are roughly equivalent.

[7] To see this, notice that for a given $\mathtt{c}$, the total contribution to the completion likelihood from all groundings in its formula group is $\sum_{\mathtt{f}} w_{\mathtt{c},\mathtt{f}} n_{\mathtt{c},\mathtt{f}}$. In addition, $\sum_{\mathtt{f}} n_{\mathtt{c},\mathtt{f}} = n_{\mathtt{c}}$ and there is the local normalization constraint $\sum_{\mathtt{f}} e^{w_{\mathtt{c},\mathtt{f}}} = 1$. The optimal weights $w_{\mathtt{c},\mathtt{f}}$ are easily derived by solving this constrained optimization problem.

Algorithm 2 USP-Learn(*MLN, QLFs*)

```
Create initial clusters and semantic parses
Merge clusters with the same core form
Agenda ← ∅
repeat
  for all candidate operations O do
    Score O by log-likelihood improvement
    if score is above a threshold then
      Add O to agenda
    end if
  end for
  Execute the highest scoring operation O* in the agenda
  Regenerate MAP parses for affected QLFs and update agenda and candidate operations
until agenda is empty
return the MLN with learned weights and the semantic parses
```

Another major challenge in USP learning is the summation in the likelihood, which is over all possible semantic parses for a given dependency tree. Even an efficient sampler like MC-SAT (Poon and Domingos, 2006), as used in Poon & Domingos (2008), would have a hard time generating accurate estimates within a reasonable amount of time. On the other hand, as already noted in the previous section, the lambda-form distribution is generally sparse. Large lambda-forms are rare, as they correspond to complex expressions that are often decomposable into smaller ones. Moreover, while ambiguities are present at the lexical level, they quickly diminish when more words are present. Therefore, a lambda form can usually only belong to a small number of clusters, if not a unique one. We thus simplify the problem by approximating the sum with the mode, and search instead for the L and θ that maximize $\log P_\theta(Q, L)$. Since the optimal weights and log-likelihood can be derived in closed form given the semantic parses L, we simply search over semantic parses, evaluating them using log-likelihood.

Algorithm 2 gives pseudo-code for our algorithm. The input consists of an MLN without weights and the QLFs for the training sentences. Two operators are used for updating semantic parses. The first is to merge two clusters, denoted by $\mathtt{MERGE}(\mathtt{C}_1, \mathtt{C}_2)$ for clusters $\mathtt{C}_1, \mathtt{C}_2$, which does the following:

1. Create a new cluster $\mathtt{C}$ and add all core forms in $\mathtt{C}_1, \mathtt{C}_2$ to $\mathtt{C}$;
2. Create new argument types for $\mathtt{C}$ by merging those in $\mathtt{C}_1, \mathtt{C}_2$ so as to maximize the log-likelihood;
3. Remove $\mathtt{C}_1, \mathtt{C}_2$.

Here, merging two argument types refers to pooling their argument forms to create a new argument type. Enumerating all possible ways of creating new argument types is intractable. USP approximates it by considering one type at a time and either creating a new type for it or merging it to types already considered, whichever maximizes the log-likelihood. The types are considered in decreasing order of their numbers of occurrences so that more information is available for each decision. MERGE clusters syntactically different expressions whose meanings appear to be the same according to the model.

The second operator is to create a new cluster by composing two existing ones, denoted by $\mathtt{COMPOSE}(\mathtt{C}_\mathtt{R}, \mathtt{C}_\mathtt{A})$, which does the following:

1. Create a new cluster $\mathtt{C}$;
2. Find all parts $\mathtt{r} \in \mathtt{C}_\mathtt{R}, \mathtt{a} \in \mathtt{C}_\mathtt{A}$ such that $\mathtt{a}$ is an argument of $\mathtt{r}$, compose them to $\mathtt{r}(\mathtt{a})$ by λ-reduction and add the new part to $\mathtt{C}$;
3. Create new argument types for $\mathtt{C}$ from the argument forms of $\mathtt{r}(\mathtt{a})$ so as to maximize the log-likelihood.

COMPOSE creates clusters of large lambda-forms if they tend to be composed of the same subforms (e.g., the lambda form for "is next to"). These lambda-forms may later be merged with other clusters (e.g., borders).

At learning time, USP maintains an *agenda* that contains operations that have been evaluated and are pending execution. During initialization, USP forms a part and creates a new cluster for each unary atom $\mathtt{u}(\mathtt{n})$. It also assigns binary atoms of the form $\mathtt{b}(\mathtt{n}, \mathtt{n}')$ to the part as argument forms and creates a new argument type for each. This forms the initial clustering and semantic parses. USP then merges clusters with the same core form (i.e., the same unary predicate) using MERGE.[8] At each step, USP evaluates the candidate operations and adds them to the agenda if the improvement is above a threshold.[9] The operation with the highest score is executed, and the parameters are updated with the new optimal values. The QLFs which contain an affected part are reparsed, and operations in the agenda whose score might be affected are re-evaluated. These changes are done very efficiently using inverted indexes. We omit the details here due to space limitations. USP terminates when the agenda is empty, and outputs the current MLN parameters and semantic parses.

USP learning uses the same optimization objective as hard EM, and is also guaranteed to find a local optimum since at each step it improves the log-likelihood. It differs from EM in directly optimizing the likelihood instead of a lower bound.

6 Experiments

6.1 Task

Evaluating unsupervised semantic parsers is difficult, because there is no predefined formal language or gold logical forms for the input sentences. Thus the best way to test them is by using them for the ultimate goal: answering questions based on the input corpus. In this paper, we applied USP to extracting knowledge from biomedical abstracts and evaluated its performance in answering a set of questions that simulate the information needs of biomedical researchers. We used the GENIA dataset (Kim et al., 2003) as the source for knowledge extraction. It contains 1999 PubMed abstracts and marks all mentions of biomedical entities according to the GENIA ontology, such as cell, protein, and DNA. As a first approximation to the questions a biomedical researcher might ask, we generated a set of two thousand questions on relations between entities. Sample questions are: "What regulates MIP-1alpha?", "What does anti-STAT 1 inhibit?". To simulate the real information need, we sample the relations from the 100 most frequently used verbs (excluding the auxiliary verbs *be*, *have*, and *do*), and sample the entities from those annotated in GENIA, both according to their numbers of occurrences. We evaluated USP by the number of answers it provided and the accuracy as determined by manual labeling.[10]

[8] Word-sense disambiguation can be handled by including a new kind of operator that splits a cluster into subclusters. We leave this to future work.

[9] We currently set it to 10 to favor precision and guard against errors due to inexact estimates.

[10] The labels and questions are available at http://alchemy.cs.washington.edu/papers/poon09.

6.2 Systems

Since USP is the first unsupervised semantic parser, conducting a meaningful comparison of it with other systems is not straightforward. Standard question-answering (QA) benchmarks do not provide the most appropriate comparison, because they tend to simultaneously emphasize other aspects not directly related to semantic parsing. Moreover, most state-of-the-art QA systems use supervised learning in their key components and/or require domain-specific engineering efforts. The closest available system to USP in aims and capabilities is TextRunner (Banko et al., 2007), and we compare with it. TextRunner is the state-of-the-art system for open-domain information extraction; its goal is to extract knowledge from text without using supervised labels. Given that a central challenge to semantic parsing is resolving syntactic variations of the same meaning, we also compare with RESOLVER (Yates and Etzioni, 2009), a state-of-the-art unsupervised system based on TextRunner for jointly resolving entities and relations, and DIRT (Lin and Pantel, 2001), which resolves paraphrases of binary relations. Finally, we also compared to an informed baseline based on keyword matching.

Keyword: We consider a baseline system based on keyword matching. The question substring containing the verb and the available argument is directly matched with the input text, ignoring case and morphology. We consider two ways to derive the answer given a match. The first one (**KW**) simply returns the rest of sentence on the other side of the verb. The second one (**KW-SYN**) is informed by syntax: the answer is extracted from the subject or object of the verb, depending on the question. If the verb does not contain the expected argument, the sentence is ignored.

TextRunner: TextRunner inputs text and outputs relational triples in the form (R, A_1, A_2), where R is the relation string, and A_1, A_2 the argument strings. Given a triple and a question, we first match their relation strings, and then match the strings for the argument that is present in the question. If both match, we return the other argument string in the triple as an answer. We report results when exact match is used (**TR-EXACT**), or when the triple string can contain the question one as a substring (**TR-SUB**).

RESOLVER: RESOLVER (Yates and Etzioni, 2009) inputs TextRunner triples and collectively resolves coreferent relation and argument strings. On the GENIA data, using the default parameters, RESOLVER produces only a few trivial relation clusters and no argument clusters. This is not surprising, since RESOLVER assumes high redundancy in the data, and will discard any strings with fewer than 25 extractions. For a fair comparison, we also ran RESOLVER using all extractions, and manually tuned the parameters based on eyeballing of clustering quality. The best result was obtained with 25 rounds of execution and with the entity multiple set to 200 (the default is 30). To answer questions, the only difference from TextRunner is that a question string can match any string in its cluster. As in TextRunner, we report results for both exact match (**RS-EXACT**) and substring (**RS-SUB**).

DIRT: The DIRT system inputs a path and returns a set of similar paths. To use DIRT in question answering, we queried it to obtain similar paths for the relation of the question, and used these paths while matching sentences. We first used MINIPAR (Lin, 1998) to parse input text using the same dependencies as DIRT. To determine a match, we first check if the sentence contains the question path or one of its DIRT paths. If so, and if the available argument slot in the question is contained in the one in the sentence, it is a match, and we return the other argument slot from the sentence if it is present. Ideally, a fair comparison will require running DIRT on the GENIA text, but we were not able to obtain the source code. We thus resorted to using the latest DIRT database released by the author, which contains paths extracted from a large corpus with more than 1GB of text. This puts DIRT in a very advantageous position compared with other systems. In our experiments, we used the top three similar paths, as including more results in very low precision.

USP: We built a system for knowledge extraction and question answering on top of USP. It generated Stanford dependencies (de Marneffe et al., 2006) from the input text using the Stanford parser, and then fed these to USP-Learn[11], which produced an MLN with learned weights and the MAP semantic parses of the input sentences. These MAP parses formed our knowledge base (KB). To answer questions, the system first parses the questions[12] using USP-Parse with the

[11] α and β are set to -5 and -10.

[12] The question slot is replaced by a dummy word.

Table 1: Comparison of question answering results on the GENIA dataset.

	# Total	# Correct	Accuracy
KW	150	67	45%
KW-SYN	87	67	77%
TR-EXACT	29	23	79%
TR-SUB	152	81	53%
RS-EXACT	53	24	45%
RS-SUB	196	81	41%
DIRT	159	94	59%
USP	**334**	**295**	**88%**

learned MLN, and then matches the question parse to parses in the KB by testing subsumption (i.e., a question parse matches a KB one iff the former is subsumed by the latter). When a match occurs, our system then looks for arguments of type in accordance with the question. For example, if the question is "What regulates MIP-1alpha?", it searches for the argument type of the relation that contains the argument form "nsubj" for subject. If such an argument exists for the relation part, it will be returned as the answer.

6.3 Results

Table 1 shows the results for all systems. USP extracted the highest number of answers, almost doubling that of the second highest (RS-SUB). It obtained the highest accuracy at 88%, and the number of correct answers it extracted is three times that of the second highest system. The informed baseline (KW-SYN) did surprisingly well compared to systems other than USP, in terms of accuracy and number of correct answers. TextRunner achieved good accuracy when exact match is used (TR-EXACT), but only obtained a fraction of the answers compared to USP. With substring match, its recall substantially improved, but precision dropped more than 20 points. RESOLVER improved the number of extracted answers by sanctioning more matches based on the clusters it generated. However, most of those additional answers are incorrect due to wrong clustering. DIRT obtained the second highest number of correct answers, but its precision is quite low because the similar paths contain many errors.

6.4 Qualitative Analysis

Manual inspection shows that USP is able to resolve many nontrivial syntactic variations without user supervision. It consistently resolves the syntactic difference between active and passive voices. It successfully identifies many distinct argument forms that mean the same (e.g., "X stimulates Y" $\approx$ "Y is stimulated with X", "expression of X" $\approx$ "X expression"). It also resolves many nouns correctly and forms meaningful groups of relations. Here are some sample clusters in core forms:

{investigate, examine, evaluate, analyze, study, assay}

{diminish, reduce, decrease, attenuate}

{synthesis, production, secretion, release}

{dramatically, substantially, significantly}

An example question-answer pair, together with the source sentence, is shown below:

Q: What does IL-13 enhance?

A: The 12-lipoxygenase activity of murine macrophages.

Sentence: The data presented here indicate that (1) the 12-lipoxygenase activity of murine macrophages is upregulated in vitro and in vivo by IL-4 and/or IL-13, . . .

7 Conclusion

This paper introduces the first unsupervised approach to learning semantic parsers. Our USP system is based on Markov logic, and recursively clusters expressions to abstract away syntactic variations of the same meaning. We have successfully applied USP to extracting a knowledge base from biomedical text and answering questions based on it.

Directions for future work include: better handling of antonyms, subsumption relations among expressions, quantifier scoping, more complex lambda forms, etc.; use of context and discourse to aid expression clustering and semantic parsing; more efficient learning and inference; application to larger corpora; etc.

8 Acknowledgements

We thank the anonymous reviewers for their comments. This research was partly funded by ARO grant W911NF-08-1-0242, DARPA contracts FA8750-05-2-0283, FA8750-07-D-0185, HR0011-06-C-0025, HR0011-07-C-0060 and NBCH-D030010, NSF grants IIS-0534881 and IIS-0803481, and ONR grant N00014-08-1-0670. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, DARPA, NSF, ONR, or the United States Government.

References

G. Bakir, T. Hofmann, B. B. Schölkopf, A. Smola, B. Taskar, S. Vishwanathan, and (eds.). 2007. *Predicting Structured Data*. MIT Press, Cambridge, MA.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 2670–2676, Hyderabad, India. AAAI Press.

Xavier Carreras and Luis Marquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning*, pages 89–97, Boston, MA. ACL.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 449–454, Genoa, Italy. ELRA.

Ruifang Ge and Raymond J. Mooney. 2009. Learning a compositional semantic parser using an existing syntactic parser. In *Proceedings of the Forty Seventh Annual Meeting of the Association for Computational Linguistics*, Singapore. ACL.

Lise Getoor and Ben Taskar, editors. 2007. *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.

Pekka Kilpelainen. 1992. *Tree Matching Problems with Applications to Structured Text databases*. Ph.D. Thesis, Department of Computer Science, University of Helsinki.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19:180–82.

Stanley Kok and Pedro Domingos. 2008. Extracting semantic networks from text via relational clustering. In *Proceedings of the Nineteenth European Conference on Machine Learning*, pages 624–639, Antwerp, Belgium. Springer.

Percy Liang and Dan Klein. 2008. Analyzing the errors of unsupervised learning. In *Proceedings of the Forty Sixth Annual Meeting of the Association for Computational Linguistics*, pages 879–887, Columbus, OH. ACL.

Dekang Lin and Patrick Pantel. 2001. DIRT - discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 323–328, San Francisco, CA. ACM Press.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*, Granada, Spain. ELRA.

Saif Mohammad, Bonnie Dorr, and Graeme Hirst. 2008. Computing word-pair antonymy. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 982–991, Honolulu, HI. ACL.

Raymond J. Mooney. 2007. Learning for semantic parsing. In *Proceedings of the Eighth International Conference on Computational Linguistics and Intelligent Text Processing*, pages 311–324, Mexico City, Mexico. Springer.

Hoifung Poon and Pedro Domingos. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the Twenty First National Conference on Artificial Intelligence*, pages 458–463, Boston, MA. AAAI Press.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 649–658, Honolulu, HI. ACL.

M. Richardson and P. Domingos. 2006. Markov logic networks. *Machine Learning*, 62:107–136.

Alexander Yates and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34:255–296.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammers. In *Proceedings of the Twenty First Conference on Uncertainty in Artificial Intelligence*, pages 658–666, Edinburgh, Scotland. AUAI Press.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 878–887, Prague, Czech. ACL.

Graph Alignment for Semi-Supervised Semantic Role Labeling

Hagen Fürstenau
Dept. of Computational Linguistics
Saarland University
Saarbrücken, Germany
hagenf@coli.uni-saarland.de

Mirella Lapata
School of Informatics
University of Edinburgh
Edinburgh, UK
mlap@inf.ed.ac.uk

Abstract

Unknown lexical items present a major obstacle to the development of broad-coverage semantic role labeling systems. We address this problem with a semi-supervised learning approach which acquires training instances for unseen verbs from an unlabeled corpus. Our method relies on the hypothesis that unknown lexical items will be structurally and semantically similar to known items for which annotations are available. Accordingly, we represent known and unknown sentences as graphs, formalize the search for the most similar verb as a graph alignment problem and solve the optimization using integer linear programming. Experimental results show that role labeling performance for unknown lexical items improves with training data produced automatically by our method.

1 Introduction

Semantic role labeling, the task of automatically identifying the semantic roles conveyed by sentential constituents, has recently attracted much attention in the literature. The ability to express the relations between predicates and their arguments while abstracting over surface syntactic configurations holds promise for many applications that require broad coverage semantic processing. Examples include information extraction (Surdeanu et al., 2003), question answering (Narayanan and Harabagiu, 2004), machine translation (Boas, 2005), and summarization (Melli et al., 2005).

Much progress in the area of semantic role labeling is due to the creation of resources like FrameNet (Fillmore et al., 2003), which document the surface realization of semantic roles in real world corpora. Such data is paramount for developing semantic role labelers which are usually based on supervised learning techniques and thus require training on role-annotated data. Examples of the training instances provided in FrameNet are given below:

(1) a. If [you]$_{Agent}$ [carelessly]$_{Manner}$ chance going back there, you deserve what you get.

b. Only [one winner]$_{Buyer}$ purchased [the paintings]$_{Goods}$

c. [Rachel]$_{Agent}$ injured [her friend]$_{Victim}$ [by closing the car door on his left hand]$_{Means}$.

Each verb in the example sentences evokes a *frame* which is situation-specific. For instance, *chance* evokes the *Daring* frame, *purchased* the *Commerce_buy* frame, and *injured* the *Cause_harm* frame. In addition, frames are associated with semantic roles corresponding to salient entities present in the situation evoked by the predicate. The semantic roles for the frame *Daring* are *Agent* and *Manner*, whereas for *Commerce_buy* these are *Buyer* and *Goods*. A system trained on large amounts of such *hand-annotated* sentences typically learns to identify the boundaries of the arguments of the verb predicate (argument identification) and label them with semantic roles (argument classification).

A variety of methods have been developed for semantic role labeling with reasonably good performance (F_1 measures in the low 80s on standard test collections for English; we refer the interested reader to the proceedings of the SemEval-2007 shared task (Baker et al., 2007) for an overview of the state-of-the-art). Unfortunately, the reliance on training data, which is both difficult and highly expensive to produce, presents a major obstacle to the widespread application of semantic role labeling across different languages and text genres. The English FrameNet (version 1.3) is not

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 11–20,
Singapore, 6-7 August 2009.

a small resource — it contains 502 frames covering 5,866 lexical entries and 135,000 annotated sentences. Nevertheless, by virtue of being under development it is incomplete. Lexical items (i.e., predicates evoking existing frames) are missing as well as frames and annotated sentences (their number varies greatly across lexical items). Considering how the performance of supervised systems degrades on out-of-domain data (Baker et al., 2007), not to mention unseen events, semi-supervised or unsupervised methods seem to offer the primary near-term hope for broad coverage semantic role labeling.

In this work, we develop a semi-supervised method for enhancing FrameNet with additional annotations which could then be used for classifier training. We assume that an initial set of labeled examples is available. Then, faced with an unknown predicate, i.e., a predicate that does not evoke any frame according to the FrameNet database, we must decide (a) which frames it belongs to and (b) how to automatically annotate example sentences containing the predicate. We solve both problems jointly, using a graph alignment algorithm. Specifically, we view the task of inferring annotations for new verbs as an instance of a structural matching problem and follow a graph-based formulation for pairwise global network alignment (Klau, 2009). Labeled and unlabeled sentences are represented as dependency-graphs; we formulate the search for an optimal alignment as an integer linear program where different graph alignments are scored using a function based on semantic and structural similarity. We evaluate our algorithm in two ways. We assess how accurate it is in predicting the frame for an unknown verb and also evaluate whether the annotations we produce are useful for semantic role labeling.

In the following section we provide an overview of related work. Next, we describe our graph-alignment model in more detail (Section 3) and present the resources and evaluation methodology used in our experiments (Section 4). We conclude the paper by presenting and discussing our results.

2 Related Work

Much previous work has focused on creating FrameNet-style annotations for languages other than English. A common strategy is to exploit parallel corpora and transfer annotations from English sentences onto their translations (Padó and Lapata, 2006; Johansson and Nugues, 2006). Other work attempts to automatically augment the English FrameNet in a monolingual setting either by extending its coverage or by creating additional training data.

There has been growing interest recently in determining the frame membership for unknown predicates. This is a challenging task, FrameNet currently lists 502 frames with example sentences which are simply too many (potentially related) classes to consider for a hypothetical system. Moreover, predicates may have to be assigned to multiple frames, on account of lexical ambiguity. Previous work has mainly used WordNet (Fellbaum, 1998) to extend FrameNet. For example, Burchardt et al. (2005) apply a word sense disambiguation system to annotate predicates with a WordNet sense and hyponyms of these predicates are then assumed to evoke the same frame. Johansson and Nugues (2007) treat this problem as an instance of supervised classification. Using a feature representation based also on WordNet, they learn a classifier for each frame which decides whether an unseen word belongs to the frame or not. Pennacchiotti et al. (2008) create "distributional profiles" for frames. Each frame is represented as a vector, the (weighted) centroid of the vectors representing the meaning of the predicates it evokes. Unknown predicates are then assigned to the most similar frame. They also propose a WordNet-based model that computes the similarity between the synsets representing an unknown predicate and those activated by the predicates of a frame.

All the approaches described above are type-based. They place more emphasis on extending the lexicon rather than the annotations that come with it. In our earlier work (Fürstenau and Lapata, 2009) we acquire new training instances, by projecting annotations from existing FrameNet sentences to new unseen ones. The proposed method is token-based, however, it only produces annotations for known verbs, i.e., verbs that FrameNet lists as evoking a given frame.

In this paper we generalize the proposals of Pennacchiotti et al. (2008) and Fürstenau and Lapata (2009) in a unified framework. We create training data for semantic role labeling of unknown predicates by projection of annotations from labeled onto unlabeled data. This projection is con-

ceptualized as a graph alignment problem where we seek to find a globally optimal alignment subject to semantic and structural constraints. Instead of predicting the same frame for each occurence of an unknown predicate, we consider a set of candidate frames and allow projection from any labeled predicate that can evoke one of these frames. This allows us to make instance-based decisions and thus account for predicate ambiguity.

3 Graph Alignment Method

Our approach acquires annotations for an unknown frame evoking verb by selecting sentences featuring this verb from a large unlabeled corpus (the *expansion* corpus). The choice is based upon a measure of similarity between the predicate-argument structure of the unknown verb and those of similar verbs in a manually labeled corpus (the *seed* corpus). We formulate the problem of finding the most similar verbs as the search for an optimal graph alignment (we represent labeled and unlabeled sentences as dependency graphs). Conveniently, this allows us to create labeled training instances for the unknown verb by projecting role labels from the most similar seed instance. The annotations can be subsequently used for training a semantic role labeler.

Given an unknown verb, the first step is to narrow down the number of frames it could potentially evoke. FrameNet provides definitions for more than 500 frames, of which we entertain only a small number. This is done using a method similar to Pennacchiotti et al. (2008). Each frame is represented in a semantic space as the centroid of the vectors of all its known frame evoking verbs. For an unknown verb we then consider as frame candidates the k closest frames according to a measure of distributional similarity (which we compute between the unknown verb's vector and the frame centroid vector). We provide details of the semantic space we used in our experiments in Section 4.

Next, we compare each sentence featuring the unknown verb in question to labeled sentences featuring known verbs which according to FrameNet evoke any of the k candidate frames. If sufficiently similar seeds exist, the unlabeled sentence is annotated by projecting role labels from the most similar one. The similarity score of this best match is recorded as a measure of the quality (or reliability) of the new instance. After carrying out this procedure for all sentences in the expansion corpus featuring an unknown verb, we collect the highest scoring new instances and add them back to our seed corpus as new training items. In the following we discuss in more detail how the similarity of predicate-argument structures is assessed.

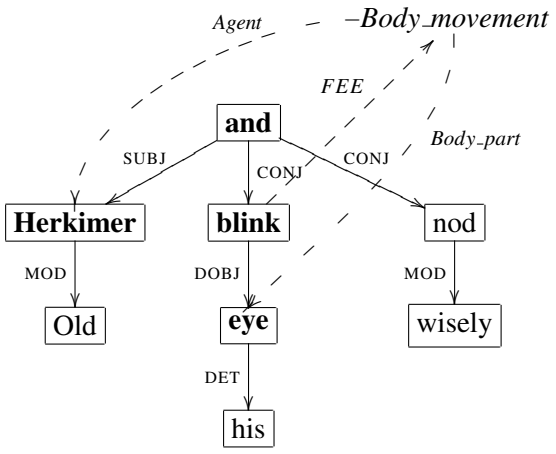


Figure 1: Annotated dependency graph for the sentence *Old Herkimer blinked his eye and nodded wisely.* The alignment domain is indicated in bold face. Labels in italics denote frame roles, whereas grammatical roles are rendered in small capitals. The verb *blink* evokes the frame *Body_Movement*.

3.1 Alignment Scoring

Let s be a semantically labeled dependency graph in which node n_{FEE} represents the frame evoking verb. Here, we use the term "labeled" to indicate that the graph contains semantic role labels in addition to grammatical role labels (e.g., subject or object). Let g be an unlabeled graph and n_{target} a verbal node in it. The "unlabeled" graph contains grammatical roles but no semantic roles. We wish to find an alignment between the predicate-argument structures of n_{FEE} and n_{target}, respectively. Such an alignment takes the form of a function σ from a set M of nodes of s (the *alignment domain*) to a set N of nodes of g (the *alignment range*). These two sets represent the relevant predicate-argument structures within the two graphs; nodes that are not members of these sets are excluded from any further computations.

If there were no mismatches between (frame) semantic arguments and syntactic arguments, we would expect all roles in s to be instantiated by syntactic dependents in n_{FEE}. This is usually the case but not always. We cannot therefore sim-

ply define M as the set of direct dependents of the predicate, but also have to consider *complex paths* between n_{FEE} and role bearing nodes. An example is given in Figure 1, where the role *Agent* is filled by a node which is not dominated by the frame evoking verb *blink*; instead, it is connected to *blink* by the complex path (CONJ^{-1}, SUBJ). For a given seed s we build a list of all such complex paths and also include all nodes of s connected to n_{FEE} by one of these paths. We thus define the alignment domain M as:

1. the predicate node n_{FEE}
2. all direct dependents of n_{FEE}, except auxiliaries
3. all nodes on complex paths originating in n_{FEE}
4. single direct dependents of any preposition or conjunction node which is in (2) or end-point of a complex path covered in (3)

The last rule ensures that the semantic heads of prepositional phrases and conjunctions are included in the alignment domain.

The alignment range N is defined in a similar way. However, we cannot extract complex paths from the unlabeled graph g, as it does not contain semantic role information. Therefore, we use the same list of complex paths extracted from s. Note that this introduces an unavoidable asymmetry into our similarity computation.

An alignment is a function $\sigma : M \rightarrow N \cup \{\varepsilon\}$ which is injective for all values except ε, i.e., $\sigma(n_1) = \sigma(n_2) \neq \varepsilon \Rightarrow n_1 = n_2$. We score the similarity of two subgraphs expressed by an alignment function σ by the following term:

$$\sum_{\substack{n \in M \\ \sigma(n) \neq \varepsilon}} \text{sem}(n, \sigma(n)) + \alpha \sum_{\substack{(n_1,n_2) \in E(M) \\ (\sigma(n_1),\sigma(n_2)) \in E(N)}} \text{syn}\left(r_{n_2}^{n_1}, r_{\sigma(n_2)}^{\sigma(n_1)}\right) \quad (2)$$

Here, sem represents a semantic similarity measure between graph nodes and syn a syntactic similarity measure between the grammatical role labels of graph edges. $E(M)$ and $E(N)$ are the sets of all graph edges between nodes of M and nodes of N, respectively, and $r_{n_2}^{n_1}$ denotes the grammatical relation between nodes n_1 and n_2.

Equation (2) expresses the similarity between two predicate-argument structures in terms of the sum of semantic similarity scores of aligned graph nodes and the sum of syntactic similarity scores of aligned graph edges. The relative weight of these two sums is determined by the parameter α. Figure 2 shows an example of an alignment between two dependency graphs. Here, the aligned node pairs *thud* and *thump*, *back* and *rest*, *against* and *against*, as well as *wall* and *front* contribute semantic similarity scores, while the three edge pairs SUBJ and SUBJ, IOBJ and IOBJ, as well as DOBJ and DOBJ contribute syntactic similarity scores.

We normalize the resulting score so that it always falls within the interval $[0, 1]$. To take into account unaligned nodes in both the alignment domain and the alignment range, we divide Equation (2) by:

$$\sqrt{|M| \cdot |N|} + \alpha \sqrt{|E(M)| \cdot |E(N)|} \quad (3)$$

A trivial alignment of a seed with itself where all semantic and syntactic scores are 1 will thus receive a score of:

$$\frac{|M| \cdot 1 + \alpha \cdot |E(M)| \cdot 1}{\sqrt{|M|^2} + \alpha \sqrt{E(M)^2}} = 1 \quad (4)$$

which is the largest possible similarity score. The lowest possible score is obviously 0, assuming that the semantic and syntactic scores cannot be negative.

Considerable latitude is available in selecting the semantic and syntactic similarity measures. With regard to semantic similarity, WordNet is a prime contender and indeed has been previously used to acquire new predicates in FrameNet (Pennacchiotti et al., 2008; Burchardt et al., 2005; Johansson and Nugues, 2007). Syntactic similarity may be operationalized in many ways, for example by taking account a hierarchy of grammatical relations (Keenan and Comrie, 1977). Our experiments employed relatively simple instantiations of these measures. We did not make use of WordNet, as we were interested in exploring the setting where WordNet is not available or has limited coverage. Therefore, we approximate the semantic similarity between two nodes via distributional similarity. We present the details of the semantic space model we used in Section 4.

If n and n' are both nouns, verbs or adjectives, we set:

$$\text{sem}(n, n') := \cos(\vec{v}_n, \vec{v}_{n'}) \quad (5)$$

where $\vec{v}_n$ and $\vec{v}_{n'}$ are the vectors representing the lemmas of n and n' respectively. If n and n'

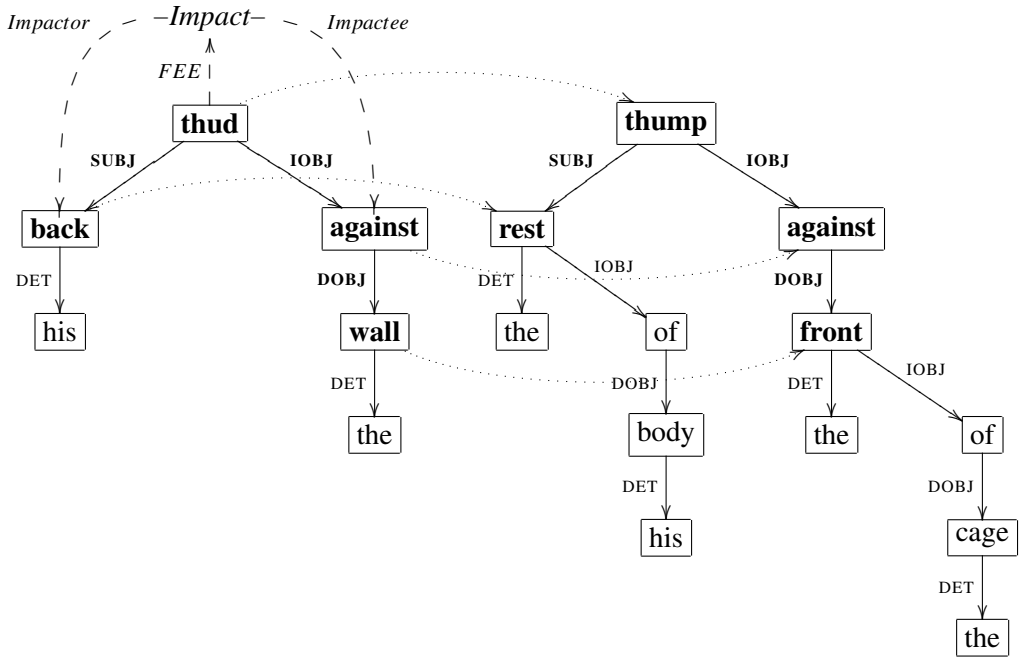


Figure 2: The dotted arrows show aligned nodes in the graphs for the two sentences *His back thudded against the wall.* and *The rest of his body thumped against the front of the cage.* (Graph edges are also aligned to each other.) The alignment domain and alignment range are indicated in bold face. The verb *thud* evokes the frame *Impact*.

are identical prepositions or conjunctions we set $\text{sem}(n,n') := 1$. In all other cases $\text{sem}(n,n') := 0$. As far as syntactic similarity is concerned, we chose the simplest metric possible and set:

$$\text{syn}\left(r,r'\right) := \begin{cases} 1 & \text{if } r = r' \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

3.2 Alignment Search

The problem of finding the best alignment according to the scoring function presented in Equation (2) can be formulated as an integer linear program. Let the binary variables x_{ik} indicate whether node n_i of graph s is aligned to node n_k of graph g. Since it is not only nodes but also graph edges that must be aligned we further introduce binary variables y_{ijkl}, where $y_{ijkl} = 1$ indicates that the edge between nodes n_i and n_j of graph s is aligned to the edge between nodes n_k and n_l of graph g. This follows a general formulation of the graph alignment problem based on maximum structural matching (Klau, 2009). In order for the x_{ik} and y_{ijkl} variables to represent a valid alignment, the following constraints must hold:

1. Each node of s is aligned to at most one node of g: $\sum_k x_{ik} \leq 1$
2. Each node of g is aligned to at most one node of s: $\sum_i x_{ik} \leq 1$
3. Two edges may only be aligned if their adjacent nodes are aligned: $y_{ijkl} \leq x_{ik}$ and $y_{ijkl} \leq x_{jl}$

The scoring function then becomes:

$$\sum_{i,k} \text{sem}(n_i,n_k)x_{ik} + \alpha \cdot \sum_{i,j,k,l} \text{syn}\left(r^{n_i}_{n_j}, r^{n_k}_{n_l}\right) y_{ijkl} \quad (7)$$

We solve this optimization problem with a version of the branch-and-bound algorithm (Land and Doig, 1960). In general, this graph alignment problem is NP-hard (Klau, 2009) and usually solved approximately following a procedure similar to beam search. However, the special structure of constraints 1 to 3, originating from the required injectivity of the alignment function, allows us to solve the optimization exactly. Our implementation of the branch-and-bound algorithm does not generally run in polynomial time, however, we found that in practice we could efficiently compute optimal alignments in almost all cases (less than 0.1% of alignment pairs in our data could not be solved in reasonable time). This relatively benign behavior depends crucially on the fact that we do not have to consider alignments between

full graphs, and the number of nodes in the aligned subgraphs is limited.

4 Experimental Design

In this section we present our experimental set-up for assessing the performance of our method. We give details on the data sets we used, describe the baselines we adopted for comparison with our approach, and explain how our system output was evaluated.

Data Our experiments used annotated sentences from FrameNet as a seed corpus. These were augmented with automatically labeled sentences from the BNC which we used as our expansion corpus. FrameNet sentences were parsed with RASP (Briscoe et al., 2006). In addition to phrase structure trees, RASP delivers a dependency-based representation of the sentence which we used in our experiments. FrameNet role annotations were mapped onto those dependency graph nodes that corresponded most closely to the annotated substring (see Fürstenau (2008) for a detailed description of the mapping algorithm). BNC sentences were also parsed with RASP (Andersen et al., 2008).

We randomly split the FrameNet corpus[1] into 80% training set, 10% test set, and 10% development set. Next, all frame evoking verbs in the training set were ordered by their number of occurrence and split into two groups, *seen* and *unseen*. Every other verb from the ordered list was considered unseen. This quasi-random split covers a broad range of predicates with a varying number of annotations. Accordingly, the FrameNet sentences in the training and test sets were divided into the sets *train_seen*, *train_unseen*, *test_seen*, and *test_unseen*. As we explain below, this was necessary for evaluation purposes.

The *train_seen* dataset consisted of 24,220 sentences, with 1,238 distinct frame evoking verbs, whereas *train_unseen* contained 24,315 sentences with the same number of frame evoking verbs. Analogously, *test_seen* had 2,990 sentences and 817 unique frame evoking verbs; the number of sentences in *test_unseen* was 3,064 (with 847 unique frame evoking verbs).

Model Parameters The alignment model presented in Section 3 crucially relies on the similarity function that scores potential alignments (see Equation (2)). This function has a free parameter, the weight α for determining the relative contribution of semantic and syntactic similarity. We tuned α using leave-one-out cross-validation on the development set. For each annotated sentence in this set we found its most similar other sentence and determined the best alignment between the two dependency graphs representing them. Since the true annotations for each sentence were available, it was possible to evaluate the accuracy of our method for any α value. We did this by comparing the true annotation of a sentence to the annotation its nearest neighbor would have induced by projection. Following this procedure, we obtained best results with $\alpha = 0.2$.

The semantic similarity measure relies on a semantic space model which we built on a lemmatized version of the BNC. Our implementation followed closely the model presented in Fürstenau and Lapata (2009) as it was used in a similar task and obtained good results. Specifically, we used a context window of five words on either side of the target word, and 2,000 vector dimensions. These were the common context words in the BNC. Their values were set to the ratio of the probability of the context word given the target word to the probability of the context word overall. Semantic similarity was measured using the cosine of the angle between the vectors representing any two words. The same semantic space was used to create the distributional profile of a frame (which is the centroid of the vectors of its verbs). For each unknown verb, we consider the k most similar frame candidates (again similarity is measured via cosine). Our experiments explored different values of k ranging from 1 to 10.

Evaluation Our evaluation assessed the performance of a semantic frame and role labeler with and without the annotations produced by our method. The labeler followed closely the implementation described in Johansson and Nugues (2008). We extracted features from dependency parses corresponding to those routinely used in the semantic role labeling literature (see Baker et al. (2007) for an overview). SVM classifiers were trained[2] with the LIBLINEAR library (Fan et al., 2008) and learned to predict the frame name, role spans, and role labels. We followed

[1] Here, we consider only FrameNet example sentences featuring verbal predicates.

[2] The regularization parameter C was set to 0.1.

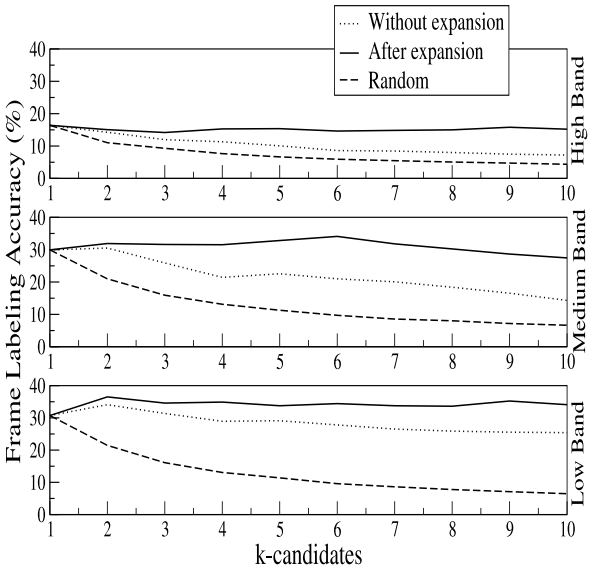


Figure 3: Frame labeling accuracy on high, medium and low frequency verbs, before and after applying our expansion method; the labeler decides among $k = 1, \ldots, 10$ candidate frames.

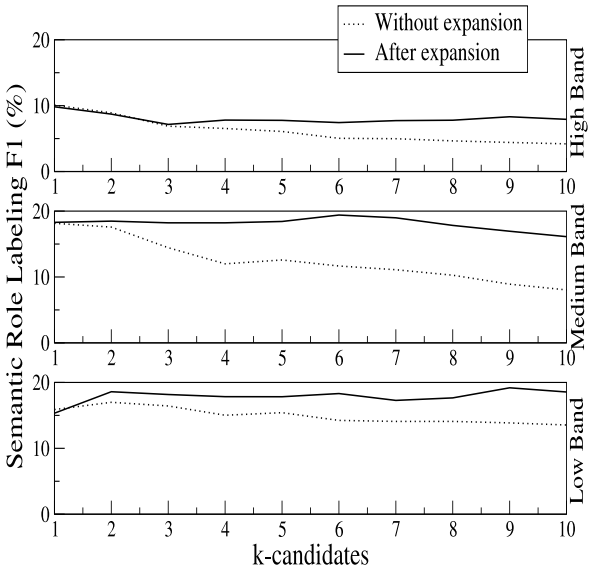


Figure 4: Role labeling F_1 for high, medium, and low frequency verbs (roles of mislabeled frames are counted as wrong); the labeler decides among $k = 1, \ldots, 10$ candidate frames.

the one-versus-one strategy for multi-class classification (Friedman, 1996).

Specifically, the labeler was trained on the *train_seen* data set without any access to training instances representative of the "unknown" verbs in *test_unseen*. We then trained the labeler on a larger set containing *train_seen* and new training examples obtained with our method. To do this, we used *train_seen* as the seed corpus and the BNC as the expansion corpus. For each "unknown" verb in *train_unseen* we obtained BNC sentences with annotations projected from their most similar seeds. The quality of these sentences as training instances varies depending on their similarity to the seed. In our experiments we added to the training set the 20 highest scoring BNC sentences per verb (adding less or more instances led to worse performance).

The average number of frames which can be evoked by a verb token in the set *test_unseen* was 1.96. About half of them (1,522 instances) can evoke only one frame, 22% can evoke two frames, and 14 instances can evoke up to 11 different frames. Finally, there are 120 instances (4%) in *test_unseen* for which the correct frame is not annotated on any sentence in *train_seen*.

5 Results

We first examine how well our method performs at frame labeling. We partitioned the frame evoking verbs in our data set into three bands (High, Medium, and Low) based on an equal division of the range of their occurrence frequency in the BNC. As frequency is strongly correlated with polysemy, the division allows us to assess how well our method is performing at different degrees of ambiguity. Figure 3 summarizes our results for High, Medium, and Low frequency verbs. The number of verbs in each band are 282, 282, and 283, respectively. We compare the frame accuracy of a labeler trained solely on the annotations available in FrameNet (Without expansion) against a labeler that also uses annotations created with our method (After expansion). Both classifiers were employed in a setting where they had to decide among k candidate frames. These were the k most similar frames to the unknown verb in question. We also show the accuracy of a simple baseline labeler, which randomly chooses one of the k candidate frames.

The graphs in Figure 3 show that for verbs in the Medium and Low frequency bands, both classifiers (with and without expansion) outperform the baseline of randomly choosing among k candidate frames. Interestingly, rather than defaulting to the most similar frame ($k = 1$), we observe that ac-

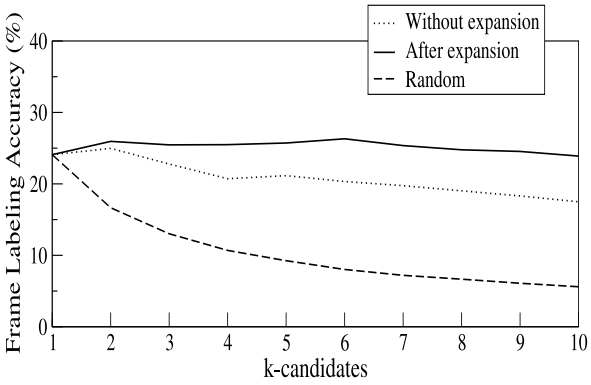


Figure 5: Hybrid frame labeling accuracy ($k = 1$ for High frequency verbs).

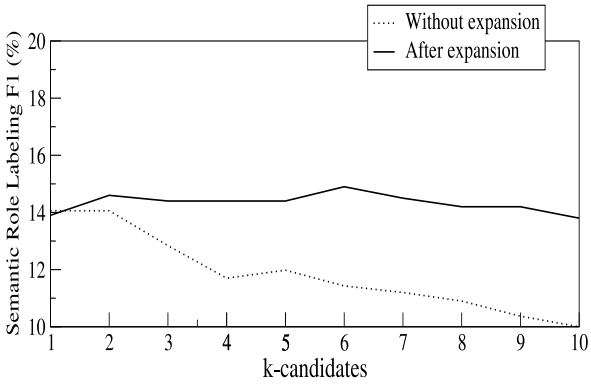


Figure 6: Hybrid role labeling F_1 ($k = 1$ for High frequency verbs).

curacy improves when frame selection is viewed as a classification task. The classifier trained on the expanded training set consistently outperforms the one trained on the original training set. While this is also true for the verbs in the High frequency band, labeling accuracy peaks at $k = 1$ and does not improve when more candidate frames are considered. This is presumably due to the skewed sense distributions of high frequency verbs, and defaulting to the most likely sense achieves relatively good performance.

Next, we evaluated our method on role labeling, again by comparing the performance of our role labeler on the expanded and original training set. Since role and frame labeling are interdependent, we count all predicted roles of an incorrectly predicted frame as wrong. This unavoidably results in low role labeling scores, but allows us to directly compare performance across different settings (e.g., different number of candidate frames, with or without expansion). Figure 4 reports labeled F_1 for verbs in the High, Medium and Low frequency bands. The results are similar to those obtained for frame labeling; the role labeler trained on the the expanded training set consistently outperforms the labeler trained on the unexpanded one. (There is no obvious baseline for role labeling, which is a complex task involving the prediction of frame labels, identification of the role bearing elements, and assignment of role labels.) Again, for High frequency verbs simply defaulting to $k = 1$ performs best.

Taken together, our results on frame and role labeling indicate that our method is not very effective for High frequency verbs (which in practice should be still annotated manually). We therefore also experimented with a hybrid approach that lets the classifier choose among k candidates for Medium and Low frequency verbs and defaults to the most similar candidate for High frequency verbs. Results for this approach are shown in Figures 5 and 6. All differences between the expanded and the unexpanded classifier when choosing between the same $k > 1$ candidates are significant according to McNemar's test ($p < .05$). The best frame labeling accuracy (26.3%) is achieved by the expanded classifier when deciding among $k = 6$ candidate frames. This is significantly better ($p < .01$) than the best performance of the unexpanded classifier (25.0%), which is achieved at $k = 2$. Role labeling results follow a similar pattern. The best expanded classifier (F_1=14.9% at $k = 6$) outperforms the best unexpanded one (F_1=14.1% at $k = 2$). The difference in performance as significant at $p < 0.05$, using stratified shuffling (Noreen, 1989).

6 Conclusions

This paper presents a novel semi-supervised approach for reducing the annotation effort involved in creating resources for semantic role labeling. Our method acquires training instances for unknown verbs (i.e., verbs that are not evoked by existing FrameNet frames) from an unlabeled corpus. A key assumption underlying our work is that verbs with similar meanings will have similar argument structures. Our task then amounts to finding the seen instances that resemble the unseen instances most, and projecting their annotations. We represent this task as a graph alignment problem, and formalize the search for an optimal alignment as an integer linear program under an

objective function that takes semantic and structural similarity into account.

Experimental results show that our method improves frame and role labeling accuracy, especially for Medium and Low frequency verbs. The overall frame labeling accuracy may seem low. There are at least two reasons for this. Firstly, the unknown verb might have a frame for which no manual annotation exists. And secondly, many errors are due to near-misses, i.e., we assign the unknown verb a wrong frame which is nevertheless very similar to the right one. In this case, accuracy will not give us any credit.

An obvious direction for future work concerns improving our scoring function. Pennacchiotti et al. (2008) show that WordNet-based similarity measures outperform their simpler distributional alternatives. An interesting question is whether the incorporation of WordNet-based similarity would lead to similar improvements in our case. Also note that currently our method assigns unknown lexical items to existing frames. A better alternative would be to decide first whether the unknown item can be classified at all (because it evokes a known frame) or whether it represents a genuinely novel frame for which manual annotation must be provided.

Acknowledgments The authors acknowledge the support of DFG (IRTG 715) and EPSRC (grant GR/T04540/01). We are grateful to Richard Johansson for his help with the re-implementation of his semantic role labeler. Special thanks to Manfred Pinkal for valuable feedback on this work.

References

Øistein E. Andersen, Julien Nioche, Ted Briscoe, and John Carroll. 2008. The BNC Parsed with RASP4UIMA. In *Proceedings of the 6th International Language Resources and Evaluation Conference*, pages 865–869, Marrakech, Morocco.

Collin F. Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval-2007 Task 19: Frame Semantic Structure Extraction. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 99–104, Prague, Czech Republic.

Hans C. Boas. 2005. Semantic frames as interlingual representations for multilingual lexical databases. *International Journal of Lexicography*, 18(4):445–478.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The Second Release of the RASP System. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 77–80, Sydney, Australia.

Aljoscha Burchardt, Katrin Erk, and Anette Frank. 2005. A WordNet Detour to FrameNet. In *Proceedings of the GLDV 200 Workshop GermaNet II*, Bonn, Germany.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874.

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

Charles J. Fillmore, Christopher R. Johnson, and Miriam R. L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16:235–250.

Jerome H. Friedman. 1996. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University.

Hagen Fürstenau and Mirella Lapata. 2009. Semi-supervised semantic role labeling. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 220–228, Athens, Greece.

Hagen Fürstenau. 2008. Enriching frame semantic resources with dependency graphs. In *Proceedings of the 6th Language Resources and Evaluation Conference*, pages 1478–1484, Marrakech, Morocco.

Richard Johansson and Pierre Nugues. 2006. A FrameNet-based semantic role labeler for Swedish. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 436–443, Sydney, Australia.

Richard Johansson and Pierre Nugues. 2007. Using WordNet to extend FrameNet coverage. In Richard Johansson and Pierre Nugues, editors, *FRAME 2007: Building Frame Semantics Resources for Scandinavian and Baltic Languages*, pages 27–30, Tartu, Estonia.

Richard Johansson and Pierre Nugues. 2008. The effect of syntactic representation on semantic role labeling. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 393–400, Manchester, UK.

E. Keenan and B. Comrie. 1977. Noun phrase accessibility and universal grammar. *Linguistic Inquiry*, 8:62–100.

Gunnar W. Klau. 2009. A new graph-based method for pairwise global network alignment. *BMC Bioinformatics*, 10 (Suppl 1).

A.H. Land and A.G. Doig. 1960. An automatic method for solving discrete programming problems. *Econometrica*, 28:497–520.

Gabor Melli, Yang Wang, Yurdong Liu, Mehdi M. Kashani, Zhongmin Shi, Baohua Gu, Anoop Sarkar, and Fred Popowich. 2005. Description of SQUASH, the SFU question answering summary handler for the duc-2005 summarization task. In *Proceedings of the HLT/EMNLP Document Understanding Workshop*, Vancouver, Canada.

Srini Narayanan and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 693–701, Geneva, Switzerland.

E. Noreen. 1989. *Computer-intensive Methods for Testing Hypotheses: An Introduction*. John Wiley and Sons Inc.

Sebastian Padó and Mirella Lapata. 2006. Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1161–1168, Sydney, Australia.

Marco Pennacchiotti, Diego De Cao, Roberto Basili, Danilo Croce, and Michael Roth. 2008. Automatic induction of FrameNet lexical units. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 457–465, Honolulu, Hawaii.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 8–15, Sapporo, Japan.

Semi-supervised Semantic Role Labeling Using the Latent Words Language Model

Koen Deschacht
Department of computer science
K.U.Leuven, Belgium
koen.deschacht@cs.kuleuven.be

Marie-Francine Moens
Department of computer science
K.U.Leuven, Belgium
sien.moens@cs.kuleuven.be

Abstract

Semantic Role Labeling (SRL) has proved to be a valuable tool for performing automatic analysis of natural language texts. Currently however, most systems rely on a large training set, which is manually annotated, an effort that needs to be repeated whenever different languages or a different set of semantic roles is used in a certain application. A possible solution for this problem is semi-supervised learning, where a small set of training examples is automatically expanded using unlabeled texts. We present the Latent Words Language Model, which is a language model that learns word similarities from unlabeled texts. We use these similarities for different semi-supervised SRL methods as additional features or to automatically expand a small training set. We evaluate the methods on the PropBank dataset and find that for small training sizes our best performing system achieves an error reduction of 33.27% F1-measure compared to a state-of-the-art supervised baseline.

1 Introduction

Automatic analysis of natural language is still a very hard task to perform for a computer. Although some successful applications have been developed (see for instance (Chinchor, 1998)), implementing an automatic text analysis system is still a labour and time intensive task. Many applications would benefit from an intermediate representation of texts, where an automatic analysis is already performed which is sufficiently general to be useful in a wide range of applications.

Syntactic analysis of texts (such as Part-Of-Speech tagging and syntactic parsing) is an example of such a generic analysis, and has proved useful in applications ranging from machine translation (Marcu et al., 2006) to text mining in the bio-medical domain (Cohen and Hersh, 2005). A syntactic parse is however a representation that is very closely tied with the surface-form of natural language, in contrast to Semantic Role Labeling (SRL) which adds a layer of predicate-argument information that generalizes across different syntactic alternations (Palmer et al., 2005). SRL has received a lot of attention in the research community, and many systems have been developed (see section 2). Most of these systems rely on a large dataset for training that is manually annotated. In this paper we investigate whether we can develop a system that achieves state-of-the-art semantic role labeling without relying on a large number of labeled examples. We aim to do so by employing the Latent Words Language Model that learns *latent words* from a large unlabeled corpus. Latent words are words that (unlike observed words) did not occur at a particular position in a text, but given semantic and syntactic constraints from the context could have occurred at that particular position.

In section 2 we revise existing work on SRL and on semi-supervised learning. Section 3 outlines our supervised classifier for SRL and section 4 discusses the Latent Words Language Model. In section 5 we will combine the two models for semi-supervised role labeling. We will test the model on the standard PropBank dataset and compare it with state-of-the-art semi-supervised SRL systems in section 6 and finally in section 7 we draw conclusions and outline future work.

2 Related work

Gildea and Jurafsky (2002) were the first to describe a statistical system trained on the data from the FrameNet project to automatically assign semantic roles. This approach was soon followed by other researchers (Surdeanu et al., 2003; Pradhan et al., 2004; Xue and Palmer, 2004), focus-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 21–29,
Singapore, 6-7 August 2009.

ing on improved sets of features, improved machine learning methods or both, and SRL became a shared task at the CoNLL 2004, 2005 and 2008 conferences[1]. The best system (Johansson and Nugues, 2008) in CoNLL 2008 achieved an F1-measure of 81.65% on the workshop's evaluation corpus.

Semi-supervised learning has been suggested by many researchers as a solution to the annotation bottleneck (see (Chapelle et al., 2006; Zhu, 2005) for an overview), and has been applied successfully on a number of natural language processing tasks. Mann and McCallum (2007) apply Expectation Regularization to Named Entity Recognition and Part-Of-Speech tagging, achieving improved performance when compared to supervised methods, especially on small numbers of training data. Koo et al. (2008) present an algorithm for dependency parsing that uses clusters of semantically related words, which were learned in an unsupervised manner. There has been little research on semi-supervised learning for SRL. We refer to He and Gildea (2006) who tested active learning and co-training methods, but found little or no gain from semi-supervised learning, and to Swier and Stevenson (2004), who achieved good results using semi-supervised methods, but tested their methods on a small number of VerbNet roles, which have not been used by other SRL systems. To the best of our knowledge no system was able to reproduce the successful results of (Swier and Stevenson, 2004) on the PropBank roleset. Our approach most closely resembles the work of Fürstenau and Lapata (2009) who automatically expand a small training set using an automatic dependency alignment of unlabeled sentences. This method was tested on the FrameNet corpus and improved results when compared to a fully-supervised classifier. We will discuss their method in detail in section 5.

3 Semantic role labeling

Fillmore (1968) introduced semantic structures called semantic frames, describing abstract actions or common situations (frames) with common roles and themes (semantic roles). Inspired by this idea different resources were constructed, including FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005). An alternative approach to semantic role labeling is the framework developed by Halliday (1994) and implemented by Mehay et al. (2005). PropBank has thus far received the most attention of the research community, and is used in our work.

3.1 PropBank

The goal of the PropBank project is to add semantic information to the syntactic nodes in the English Penn Treebank. The main motivation for this annotation is the preservation of semantic roles across different syntactic realizations. Take for instance the sentences

1. The window broke.
2. John broke the window.

In both sentences the constituent "the window" is broken, although it occurs at different syntactic positions. The PropBank project defines for a large collection of verbs (excluding auxiliary verbs such as "will", "can", ...) a set of senses, that reflect the different meanings and syntactic alternations of this verb. Every sense has a number of expected roles, numbered from Arg0 to Arg5. A small number of arguments are shared among all senses of all verbs, such as temporals (Arg-TMP), locatives (Arg-LOC) and directionals (Arg-DIR). Additional to the frame definitions, PropBank has annotated a large training corpus containing approximately 113.000 annotated verbs. An example of an annotated sentence is

$$[\text{John}\ _{Arg0}][\text{broke}\ _{BREAK.01}]\ [\text{the window}\ _{Arg1}].$$

Here *BREAK*.01 is the first sense of the "break" verb. Note that (1) although roles are defined for every frame separately, in reality roles with identical names are identical or very similar for all frames, a fact that is exploited to train accurate role classifiers and (2) semantic role labeling systems typically assume that a frame is fully expressed in a single sentence and thus do not try to instantiate roles across sentence boundaries. Although the original PropBank corpus assigned semantic roles to syntactic phrases (such as noun phrases), we use the CoNLL dataset, where the PropBank corpus was converted to a dependency representation, assigning semantic roles to single (head) words.

3.2 Features

In this section we discuss the features used in the semantic role labeling system. All features but the

[1] See http://www.cnts.ua.ac.be/conll/ for an overview.

Split path feature are taken from existing semantic role labeling systems, see for example (Gildea and Jurafsky, 2002; Lim et al., 2004; Thompson et al., 2006). The number in brackets denotes the number of unique features for that type.

Word We split every sentence in (unigram) word tokens, including punctuation. (37079)

Stem We reduce the word tokens to their stem, e.g. "walks" -> "walk". (28690)

POS The part-of-speech tag for every word, e.g. "NNP" (for a singular proper noun). (77)

Neighbor POS's The concatenated part-of-speech tags of the word before and the word just after the current word, e.g. "RBS_JJR". (1787)

Path This important feature describes the path through the dependency tree from the current word to the position of the predicate, e.g. "coord↑obj↑adv↑root↓dep↓nmod↓pmod", where '↑' indicates going up a constituent and '↓' going down one constituent. (829642)

Split Path Because of the nature of the path feature, an explosion of unique features is found in a given data set. We reduce this by splitting the path in different parts and using every part as a distinct feature. We split, for example, the previous path in 6 different features: "coord", "↑obj", "↑adv", "↑root", "↓dep", "↓nmod", "↓pmod". Note that the split path feature includes the POS feature, since the first component of the path is the POS tag for the current word. This feature has not been used previously for semantic role detection. (155)

For every word w_i in the training and test set we construct the feature vector $\mathbf{f}(w_i)$, where at every position in this vector 1 indicates the presence for the corresponding feature and 0 the absence of that feature.

3.3 Discriminative model

Discriminative models have been found to outperform generative models for many different tasks including SRL (Lim et al., 2004). For this reason we also employ discriminative models here. The structure of the model was inspired by a similar (although generative) model in (Thompson et al., 2006) where it was used for semantic frame classification. The model (fig. 1) assumes that the role label r_{ij} for the word w_i is conditioned on the features $\mathbf{f}_i$ and on the role label $r_{i-1\,j}$ of the previous word and that the predicate label p_j for word w_j is conditioned on the role labels $\mathbf{R}^j$ and on the features $\mathbf{f}_j$. This model can be seen as an extension of the standard Maximum Entropy Markov Model (MEMM, see (Ratnaparkhi, 1996)) with an extra dependency on the predicate label, we will henceforth refer to this model as *MEMM+pred.*

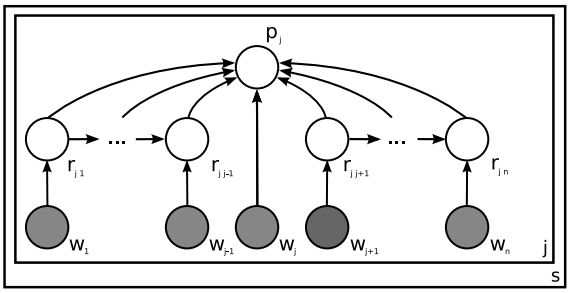


Figure 1: Discriminative model for SRL. Grey circles represent observed variables, white circles hidden variables and arrows directed dependencies. *s* ranges over all sentences in the corpus and *j* over the *n* words in the sentence.

To estimate the parameters of the *MEMM+pred* model we turn to the successful Maximum Entropy (Berger et al., 1996) parameter estimation method. The Maximum Entropy principle states that the best model given the training data is the model such that the conditional distribution defined by the model has maximum entropy subject to the constraints represented by the training examples. There is no closed form solution to find this maximum and we thus turn to an iterative method. In this work we use Generalized Iterative Scaling[2], but other methods such as (quasi-) Newton optimization could also have been used.

4 Latent Words Language Model

4.1 Rationale

As discussed in sections 1 and 3 most SRL systems are trained today on a large set of manually annotated examples. PropBank for example contains approximately 50000 sentences. This manual annotation is both time and labour-intensive, and needs to be repeated for new languages or

[2]We use the *maxent* package available on http://maxent.sourceforge.net/

for new domains requiring a different set of roles. One approach that can help to solve this problem is semi-supervised learning, where a small set of annotated examples is used together with a large set of unlabeled examples when training a SRL model.

Manual inspection of the results of the supervised model discussed in the previous section showed that the main source of errors was incorrect labeling of a word because the word token did not occur, or occurred only a small number of times in the training set. We hypothesize that knowledge of semantic similar words could overcome this problem by associating words that occurred infrequently in the training set to similar words that occurred more frequently. Furthermore, we would like to learn these similarities automatically, to be independent of knowledge sources that might not be available for all languages or domains.

The Distributional Hypothesis, supported by theoretical linguists such as Harris (1954), states that words that occur in the same contexts tend to have similar meanings. This suggests that one can learn the similarity between two words automatically by comparing their relative contexts in a large unlabeled corpus, which was confirmed by different researchers (e.g. (Lin, 1998; McDonald and Ramscar, 2001; Grefenstette, 1994)). Different methods for computing word similarities have been proposed, differing between methods to represent the context (using dependency relationship or a window of words) and between methods that, given a set of contexts, compute the similarity between different words (ranging from cosine similarity to more complex metrics such as the Jaccard index). We refer to (Lin, 1998) for a comparison of the different similarity metrics.

In the next section we propose a novel method to learn word similarities, the Latent Words Language Model (LWLM) (Deschacht and Moens, 2009). This model learns similar words and learns the a distribution over the contexts in which certain types of words occur typically.

4.2 Definition

The LWLM introduces for a text $\mathbf{T} = w_1 ... w_N$ of length N for every observed word w_i at position i a hidden variable h_i. The model is a generative model for natural language, in which the latent variable h_i is generated by its context $C(h_i)$ and the observed word w_i is generated by the latent variable h_i. In the current model we assume that the context is $C(h_i) = \mathbf{h}_{i-2}^{i-1}\mathbf{h}_{i+1}^{i+2}$ where $\mathbf{h}_{i-2}^{i-1} = h_{i-2}h_{i-1}$ is the two previous words and $\mathbf{h}_{i+1}^{i+2} = h_{i+1}h_{i+2}$ is the two next words. The observed w_i has a value from the vocabulary V, while the hidden variable h_i is unknown, and is modeled as a probability distribution over all words of V. We will see in the next section how this distribution is estimated from a large unlabeled training corpus. The aim of this model is to estimate, at every position i, a distribution for h_i, assigning high probabilities to words that are similar to w_i, given the context of this word $C(h_i)$, and low probabilities to words that are not similar to w_i in this context.

A possible interpretation of this model states that every hidden variable h_i models the "meaning" for a particular word in a particular context. In this probabilistic model, when generating a sentence, we generate the meaning of a word (which is an unobserved representation) with a certain probability, and then we generate a certain observation by writing down one of the possible words that express this meaning.

Creating a representation that models the meaning of a word is an interesting (and controversial) topic in its own right, but in this work we make the assumption that the meaning of a particular word can be modeled using other words. Modeling the meaning of a word with other words is not an unreasonable one, since it is already employed in practice by humans (e.g. by using dictionaries and thesauri) and machines (e.g. relying on a lexical resource such as WordNet) in word sense disambiguation tasks.

4.3 Parameter estimation

As we will further see the LWLM model has three probability distributions: $P(w_i|h_i)$, the probability of the observed word w_j given the latent variable h_j, $P(h_i|\mathbf{h}_{i-2}^{i-1})$, the probability of the hidden word h_j given the previous variables h_{j-2} and h_{j-1}, and $P(h_i|\mathbf{h}_{i+1}^{i+2})$, the probability of the hidden word h_j given the next variables h_{j+1} and h_{j+2}. These distributions need to be learned from a training text $\mathbf{T}_{train} =< w_0 ... w_z >$ of length Z.

4.3.1 The Baum-Welch algorithm

The attentive reader will have noticed the similarity between the proposed model and a standard second-order Hidden Markov Model (HMM) where the hidden state is dependent on the two

previous states. However, we are not able to use the standard Baum-Welch (or forward-backward) algorithm, because the hidden variable h_i is modeled as a probability distribution over all words in the vocabulary V. The Baum-Welch algorithm would result in an execution time of $O(|V|^3 NG)$ where $|V|$ is the size of the vocabulary, N is the length of the training text and G is the number of iterations needed to converge. Since in our dataset the vocabulary size is more than $30K$ words (see section 3.2), using this algorithm is not possible. Instead we use techniques of approximate inference, i.e. Gibbs sampling.

4.3.2 Initialization

Gibbs sampling starts from a random initialization for the hidden variables and then improves the estimates in subsequent iterations. In preliminary experiments it was found that a pure random initialization results in a very long burn-in-period and a poor performance of the final model. For this reason we initially set the distributions for the hidden words equal to the distribution of words as given by a standard language model[3].

4.3.3 Gibbs sampling

We store the initial estimate of the hidden variables in $\mathbf{M}^0_{train} =< h_0 ... h_Z >$, where h_i generates w_i at every position i. Gibbs sampling is a Markov Chain Monte Carlo method that updates the estimates of the hidden variables in a number of iterations. $\mathbf{M}^{\tau}_{train}$ denotes the estimate of the hidden variables in iteration τ. In every iteration a new estimate $\mathbf{M}^{\tau+1}_{train}$ is generated from the previous estimate $\mathbf{M}^{\tau}_{train}$ by selecting a random position j and updating the value of the hidden variable at that position. The probability distributions $P^{\tau}(w_j|h_j)$, $P^{\tau}(h_j|\mathbf{h}^{j-1}_{j-2})$ and $P^{\tau}(h_j|\mathbf{h}^{j+2}_{j+1})$ are constructed by collecting the counts from all positions $i \neq j$. The hidden variable h_j is dependent on h_{j-2}, h_{j-1}, h_{j+1}, h_{j+2} and w_j and we can compute the distribution of possible values for the variable h_j as

$$P^{\tau}(h_j|w_j, \mathbf{h}^{j-1}_0, \mathbf{h}^{Z}_{j+1}) = \frac{P^{\tau}(w_j|h_j)P^{\tau}(h_j|\mathbf{h}^{j-1}_{j-2}\mathbf{h}^{j+2}_{j+1})}{\sum_{h_i} P^{\tau}(w_i|h_i)P^{\tau}(h_j|\mathbf{h}^{j-1}_{j-2}\mathbf{h}^{j+2}_{j+1})}$$

We set $P(h_j|\mathbf{h}^{j-1}_{j-2}\mathbf{h}^{j+2}_{j+1}) = P(h_j|\mathbf{h}^{j-1}_{j-2}) \cdot P(h_j|\mathbf{h}^{j+2}_{j+1})$ which can be easily computed given the above distributions. We select a new value for the hidden variable according to $P^{\tau}(h_j|w_j, \mathbf{h}^{j-1}_0, \mathbf{h}^{Z}_{j+1})$ and place it at position j in $\mathbf{M}^{\tau+1}_{train}$. The current estimate for all other unobserved words remains the same. After performing this iteration a large number of times ($|V| * 10$ in this experiment), the distribution approaches the true maximum likelihood distribution. Gibbs sampling however samples this distribution, and thus will never reach it exactly. A number of iterations ($|V| * 100$) is then performed in which Gibbs sampling oscillates around the correct distribution. We collect independent samples of this distribution every $|V| * 10$ iterations, which are then used to construct the final model.

[3] We used the interpolated Kneser-Ney model as described in (Goodman, 2001).

4.4 Evaluation of the Language Model

A first evaluation of the quality of the automatically learned latent words is by translation of this model into a sequential language model and by measuring its perplexity on previously unseen texts. In (Deschacht and Moens, 2009) we perform a number of experiments, comparing different corpora (news texts from Reuters and from Associated Press, and articles from Wikipedia) and n-gram sizes (3-gram and 4-gram). We also compared the proposed model with two state-of-the-art language models, Interpolated Kneser-Ney smoothing and *fullibmpredict* (Goodman, 2001), and found that LWLM outperformed both models on all corpora, with a perplexity reduction ranging between 12.40% and 5.87%. These results show that the estimated distributions over latent words are of a high quality and lead us to believe they could be used to improve automatic text analysis, like SRL.

5 Role labeling using latent words

The previous section discussed how the LWLM learns similar words and how these similarities improved the perplexity on an unseen text of the language model derived from this model. In this section we will see how we integrate the latent words model in two novel semi-supervised SRL models and compare these with two state-of-the-art semi-supervised models for SRL and dependency parsing.

Latent words as additional features

In a first approach we estimate the distribution of latent words for every word for both the training and test set. We then use the latent words at every

position as additional probabilistic features for the discriminative model. More specifically, we append $|V|$ extra values to the feature vector $\mathbf{f}(w_j)$, containing the probability distribution over the $|V|$ possible words for the hidden variable h_i[4]. We call this the *LWFeatures* method.

This method has the advantage that it is simple to implement and that many existing SRL systems can be easily extended by adding additional features. We also expect that this method can be employed almost effortless in other information extraction tasks, such as Named Entity Recognition or Part-Of-Speech labeling.

We compare this approach to the semi-supervised method in Koo et al. (2008) who employ clusters of related words constructed by the Brown clustering algorithm (Brown et al., 1992) for syntactic processing of texts. Interestingly, this clustering algorithm has a similar objective as LWLM since it tries to optimize a class-based language model in terms of perplexity on an unseen test text. We employ a slightly different clustering method here, the *fullibmpredict* method discussed in (Goodman, 2001). This method was shown to outperform the class based model proposed in (Brown et al., 1992) and can thus be expected to discover better clusters of words. We append the feature vector $\mathbf{f}(w_j)$ with c extra values (where c is the number of clusters), respectively set to 1 if the word w_i belongs to the corresponding cluster or to 0 otherwise. We call this method the *ClusterFeatures* method.

Automatic expansion of the training set using predicate argument alignment

We compare our approach with a method proposed by Fürstenau and Lapata (2009). This approach is more tailored to the specific case of SRL and is summarized here.

Given a set of labeled seed verbs with annotated semantic roles, for every annotated verb a number of occurrences of this verb is found in unlabeled texts where the context is similar to the context of the annotated example. The context is defined here as all words in the sentence that are direct dependents of this verb, given the syntactic dependency tree. The similarity between two occurrences of a particular verb is measured by finding all different alignments $\sigma : M_\sigma \rightarrow \{1...n\}$ $(M_\sigma \subset \{1,...,m\})$ between the m dependents of the first occurrence and the n dependents of the second occurrence. Every alignment σ is assigned a score given by

$$\sum_{i \in M_\sigma} \left(A \cdot syn(g_i, g_{\sigma(i)}) + sem(w_i, w_{\sigma(i)}) - B \right)$$

where $syn(g_i, g_{\sigma(i)})$ denotes the syntactic similarity between grammatical role[5] g_i of word w_i and grammatical role $g_{\sigma(i)}$ of word $w_{\sigma(i)}$, and $sem(w_i, w_{\sigma(i)})$ measures the semantic similarity between words w_i and $w_{\sigma(i)}$. A is a constant weighting the importance of the syntactic similarity compared to semantic similarity, and B can be interpreted as the lowest similarity value for which an alignment between two arguments is possible. The syntactic similarity $syn(g_i, g_{\sigma(i)})$ is defined as 1 if the dependency relations are identical, $0 < a < 1$ if the relations are of the same type but of a different subtype[6] and 0 otherwise. The semantic similarity $sem(w_i, w_{\sigma(i)})$ is automatically estimated as the cosine similarity between the contexts of w_i and $w_{\sigma(i)}$ in a large text corpus. For details we refer to (Fürstenau and Lapata, 2009).

For every verb in the annotated training set we find the k occurrences of that verb in the unlabeled texts where the contexts are most similar given the best alignment. We then expand the training set with these examples, automatically generating an annotation using the discovered alignments. The variable k controls the trade-off between annotation confidence and expansion size. The final model is then learned by running the supervised training method on the expanded training set. We call this method *AutomaticExpansionCOS*[7]. The values for k, a, A and B are optimized automatically in every experiment on a held-out set (disjoint from both training and test set).
We adapt this approach by employing a different method for measuring semantic similarity. Given two words w_i and $w_{\sigma(i)}$ we estimate the distribution of latent words, respectively $L(h_i)$ and

[4] Probabilities smaller than $1e10^{-4}$ were set to 0 for efficiency reasons.

[5] Note that this is a syntactic role, not a semantic role as the ones discussed in this article.

[6] Subtypes are fine-grained distinctions made by the parser such as the underlying grammatical roles in passive constructions.

[7] The only major differences with (Fürstenau and Lapata, 2009) are the dependency parser which was used (the MALT parser (Nivre et al., 2006) instead of the RASP parser (Briscoe et al., 2006)) and the corpus employed to learn semantic similarities (the Reuters corpus instead of the British National Corpus). We expect that these differences will only influence the results minimally.

	5%	20%	50%	100%
Supervised	40.49%	67.23%	74.93%	78.65%
LWFeatures	**60.29%**	**72.88%**	**76.42%**	**80.98%**
ClusterFeatures	59.51%	66.70%	70.15%	72.62%
AutomaticExpansionCOS	47.05%	53.72%	64.51%	70.52%
AutomaticExpansionLW	45.40%	53.82%	65.39%	72.66%

Table 1: Results (in F1-measure) on the CoNLL 2008 test set for the different methods, comparing the supervised method (*Supervised*) with the semi-supervised methods *LWFeatures*, *ClusterFeatures*, *AutomaticExpansionCOS* and *AutomaticExpansionLW*. See section 5 for details on the different methods. Best results are in bold.

$L(h_{\sigma(i)})$. We then compute the semantic similarity measure as the Jensen-Shannon (Lin, 1997) divergence

$$JS(L(h_i)||L(h_{\sigma(i)})) = \frac{1}{2}\left[D(L(h_i)||avg) + D\left(L(h_{\sigma(i)})||avg\right)\right]$$

where $avg = (L(h_i) + L(h_{\sigma(i)}))/2$ is the average between the two distributions and $D(L(h_i)||avg)$ is the Kullback–Leiber divergence (Cover and Thomas, 2006).

Although this change might appear only a slight deviation from the original model discussed in (Fürstenau and Lapata, 2009) it is potentially an important one, since an accurate semantic similarity measure will greatly influence the accuracy of the alignments, and thus of the accuracy of the automatic expansion. We call this method *AutomaticExpansionLW*.

6 Experiments

We perform a number of experiments where we compare the fully supervised model with the semi-supervised models proposed in the previous section. We first train the LWLM model on an unlabeled 5 million word *Reuters* corpus[8].

We perform different experiments for the supervised and the four different semi-supervised methods (see previous section). Table 1 shows the results of the different methods on the test set of the CoNLL 2008 shared task. We experimented with different sizes for the training set, ranging from 5% to 100%. When using a subset of the full training set, we run 10 different experiments with random subsets and average the results.

We see that the *LWFeatures* method performs better than the other methods across all training sizes. Furthermore, these improvements are larger for smaller training sets, showing that the approach can be applied successfully in a setting where only a small number of training examples is available.

When comparing the *LWFeatures* method with the *ClusterFeatures* method we see that, although the *ClusterFeatures* method has a similar performance for small training sizes, this performance drops for larger training sizes. A possible explanation for this result is the use of the clusters employed in the *ClusterFeatures* method. By definition the clusters merge many words into one cluster, which might lead to good generalization (more important for small training sizes) but can potentially hurt precision (more important for larger training sizes).

A third observation that can be made from table 1 is that, although both automatic expansion methods (*AutomaticExpansionCOS* and *AutomaticExpansionCOS*) outperform the supervised method for the smallest training size, for other sizes of the training set they perform relatively poorly. An informal inspection showed that for some examples in the training set, little or no correct similar occurrences were found in the unlabeled text. The algorithm described in section 5 adds the most similar k occurrences to the training set for every annotated example, also for these examples where little or no similar occurrences were found. Often the automatic alignment fails to generate correct labels for these occurrences and introduces errors in the training set. In the future we would like to perform experiments that determine dynamically (for instance based on the similarity measure between occurrences) for every annotated example how many training examples to add.

[8] See http://www.daviddlewis.com/resources

7 Conclusions and future work

We have presented the Latent Words Language Model and showed how it learns, from unlabeled texts, latent words that capture the meaning of a certain word, depending on the context. We then experimented with different methods to incorporate the latent words for Semantic Role Labeling, and tested different methods on the PropBank dataset. Our best performing method showed a significant improvement over the supervised model and over methods previously proposed in the literature. On the full training set the best method performed 2.33% better than the fully supervised model, which is a 10.91% error reduction. Using only 5% of the training data the best semi-supervised model still achieved 60.29%, compared to 40.49% by the supervised model, which is an error reduction of 33.27%. These results demonstrate that the latent words learned by the LWLM help for this complex information extraction task. Furthermore we have shown that the latent words are simple to incorporate in an existing classifier by adding additional features. We would like to perform experiments on employing this model in other information extraction tasks, such as Word Sense Disambiguation or Named Entity Recognition. The current model uses the context in a very straightforward way, i.e. the two words left and right of the current word, but in the future we would like to explore more advanced methods to improve the similarity estimates. Lin (1998) for example discusses a method where a syntactic parse of the text is performed and the context of a word is modeled using dependency triples.

The other semi-supervised methods proposed here were less successful, although all improved on the supervised model for small training sizes. In the future we would like to improve the described automatic expansion methods, since we feel that their full potential has not yet been reached. More specifically we plan to experiment with more advanced methods to decide whether some automatically generated examples should be added to the training set.

Acknowledgments

The work reported in this paper was supported by the EU-IST project CLASS (Cognitive-Level Annotation using Latent Statistical Structure, IST-027978) and the IWT-SBO project AMASS++ (IWT-SBO-060051). We thank the anonymous reviewers for their helpful comments and Dennis N. Mehay for his help on clarifying the linguistic motivation of our models.

References

C.F. Baker, C.J. Fillmore, and J.B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, volume 98. Montreal, Canada.

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

T. Briscoe, J. Carroll, and R. Watson. 2006. The second release of the RASP system. In *Proceedings of the Interactive Demo Session of COLING/ACL*, volume 6.

P.F. Brown, R.L. Mercer, V.J. Della Pietra, and J.C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

O. Chapelle, B. Schölkopf, and A. Zien, editors. 2006. *Semi-Supervised Learning*. MIT Press, Cambridge, MA.

N.A. Chinchor. 1998. Overview of MUC-7/MET-2. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, volume 1.

A.M. Cohen and W.R. Hersh. 2005. A survey of current work in biomedical text mining. *Briefings in Bioinformatics*, 6(1):57–71.

T.M. Cover and J.A. Thomas. 2006. *Elements of Information Theory*. Wiley-Interscience.

Koen Deschacht and Marie-Francine Moens. 2009. The Latent Words Language Model. In *Proceedings of the 18th Annual Belgian-Dutch Conference on Machine Learning*.

C. J. Fillmore. 1968. The case for case. In E. Bach and R. Harms, editors, *Universals in Linguistic Theory*. Rinehart & Winston.

Hagen Fürstenau and Mirella Lapata. 2009. Semi-supervised semantic role labeling. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 220–228, Athens, Greece. Association for Computational Linguistics.

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

Joshua T. Goodman. 2001. A bit of progress in language modeling, extended version. Technical report, Microsoft Research.

G. Grefenstette. 1994. *Explorations in automatic thesaurus discovery*. Springer.

M.A.K. Halliday. 1994. *An Introduction to Functional Grammar (second edition)*. Edward Arnold, London.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

S. He and D. Gildea. 2006. Self-training and Co-training for Semantic Role Labeling: Primary Report. Technical report. TR 891.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with propbank and nombank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187, Manchester, England, August. Coling 2008 Organizing Committee.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 595–603.

J.-H. Lim, Y.-S. Hwang, S.-Y. Park, and H.-C. Rim. 2004. Semantic role labeling using maximum entropy model. In *Proceedings of the Eighth Conference on Computational Natural Language Learning*, pages 122–125, Boston, Massachusetts, USA. ACL.

D. Lin. 1997. Using syntactic dependency as local context to resolve word sense ambiguity. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, volume 35, pages 64–71. ACL.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational Linguistics*, pages 768–774. Association for Computational Linguistics Morristown, NJ, USA.

G.S. Mann and A. McCallum. 2007. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of the 24th International Conference on Machine Learning*, pages 593–600. ACM Press New York, USA.

D. Marcu, W. Wang, A. Echihabi, and K. Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing*, pages 44–52.

S. McDonald and M. Ramscar. 2001. Testing the distributional hypothesis: The influence of context on judgements of semantic similarity. In *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*, pages 611–616.

Dennis Mehay, Rik De Busser, and Marie-Francine Moens. 2005. Labeling generic semantic roles. In *Proceedings of the Sixth International Workshop on Computational Semantics*.

J. Nivre, J. Hall, and J. Nilsson. 2006. MaltParser: A datadriven parser-generator for dependency parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 2216–2219.

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

S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association of Computational Linguistics*, Boston, MA.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142. Association for Computational Linguistics.

M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 8–15.

R.S. Swier and S. Stevenson. 2004. Unsupervised semantic role labelling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 95–102.

C. Thompson, R. Levy, and C. Manning. 2006. A generative model for FrameNet semantic role labeling . In *Proceedings of the 14th European Conference on Machine Learning*, Cavtat-Dubrovnik, Croatia.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, volume 4.

X. Zhu. 2005. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

Semantic Dependency Parsing of NomBank and PropBank: An Efficient Integrated Approach via a Large-scale Feature Selection *

Hai Zhao(赵海)†, Wenliang Chen(陈文亮)*, Chunyu Kit†(揭春雨)
†Department of Chinese, Translation and Linguistics
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong, China
*Language Infrastructure Group, MASTAR Project
National Institute of Information and Communications Technology
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan, 619-0289
haizhao@cityu.edu.hk, chenwl@nict.go.jp

Abstract

We present an integrated dependency-based semantic role labeling system for English from both NomBank and PropBank. By introducing assistant argument labels and considering much more feature templates, two optimal feature template sets are obtained through an effective feature selection procedure and help construct a high performance single SRL system. From the evaluations on the date set of CoNLL-2008 shared task, the performance of our system is quite close to the state of the art. As to our knowledge, this is the first integrated SRL system that achieves a competitive performance against previous pipeline systems.

1 Introduction

We investigate the possibility to construct an effective integrated system for dependency-based semantic role labeling (SRL) task. This means in this work that a single system handles all these sub-tasks, predicate identification/disambiguation and argument identification/classification, regardless of whether the predicate is verbal or nominal.

Traditionally, a SRL task, either dependency or constituent based, is implemented as two subtasks, namely, argument identification and classification. If the predicate is unknown, then a predicate identification or disambiguation subtask should be additionally considered. A pipeline framework is usually adopted to handle all these sub-tasks. The reason to divide the whole task into multiple stages is two-fold, one is each subtask asks for its favorable features, the other is at the consideration of computational efficiency. Generally speaking, a joint system is slower than a pipeline system in training. (Xue and Palmer, 2004) fount out that different features suited for different sub-tasks of SRL, i.e. argument identification and classification. The results from CoNLL shared tasks in 2005 and 2008 (Carreras and Marquez, 2005; Koomen et al., 2005; Surdeanu et al., 2008; Johansson and Nugues, 2008), further show that SRL pipeline may be one of the standard to achieve a state-of-the-art performance in practice.

In the recent years, most works on SRL, including two CoNLL shared task in 2004 and 2005, focus on verbal predicates with the availability of PropBank (Palmer et al., 2005). As a complement to PropBank, NomBank (Meyers et al., 2004) annotates nominal predicates and their corresponding semantic roles using similar semantic framework as PropBank. Though SRL for nominal predicates offers more challenge, it draws relatively little attention (Jiang and Ng, 2006).

(Pustejovsky et al., 2005) discussed the issue of merging various treebanks, including PropBank, NomBank, and others. The idea of merging these two different treebanks was implemented in the CoNLL-2008 shared task (Surdeanu et al., 2008). However, few empirical studies support the necessity of an integrated learning strategy from NomBank and PropBank. Though aiming at Chinese SRL, (Xue, 2006) reported that their experiments show that simply adding the verb data to the training set of NomBank and extracting the same features from the verb and noun instances will hurt the overall performance. From the results of CoNLL-2008 shared task, the top system by (Johansson and Nugues, 2008) also used two

This study is partially supported by CERG grant 9040861 (CityU 1318/03H), CityU Strategic Research Grant 7002037.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 30–39,
Singapore, 6-7 August 2009.

different subsystems to handle verbal and nominal predicates, respectively.

Despite all the above facts, an integrated SRL system still holds some sort of merits, being easier to implement, a single-stage feature selection benefiting the whole system, an all-in-one model outputting all required semantic role information and so on.

The shared tasks at the CoNLL 2008 and 2009 are devoted to the joint learning of syntactic and semantic dependencies, which show that SRL can be well performed using only dependency syntax input. Using data and evaluation settings of the CoNLL-2008 shared task, this work will only focus on semantic dependency parsing and compares the best-performing SRL system in the CoNLL-2009 shared Task (Zhao et al., 2009b) with those in the CoNLL-2008 shared task (Surdeanu et al., 2008; Hajič et al., 2009)[1].

Aiming at main drawbacks of an integrated approach, two key techniques will be applied. 1) Assistant argument labels are introduced for the further improvement of argument pruning. This helps the development of a fast and lightweight SRL system. 2) Using a greedy feature selection algorithm, a large-scale feature engineering is performed on a much larger feature template set than that in previous work. This helps us find features that may be of benefit to all SRL sub-tasks as long as possible. As two optimal feature template sets have been proven available, for the first time we report that an integrated SRL system may provide a result close to the state-of-the-art achieved by those SRL pipelines or individual systems for some specific predicates.

2 Adaptive Argument Pruning

A word-pair classification is used to formulate semantic dependency parsing as in (Zhao and Kit, 2008). As for predicate identification or disambiguation, the first word is set as a virtual root (which is virtually set before the beginning of the sentence.) and the second as a predicate candidate. As for argument identification/classification, the first word in a word pair is specified as a predicate candidate and the second as an argument candidate. In either of case, the first word is called a semantic head, and noted as p in our feature representation, the second is called a semantic dependent and noted as a.

Word pairs are collected for the classifier in such order. The first word of the pair is set to the virtual root at first, the second word is then specified as a predicate candidate. According to the result that the predicate candidate is classified or proven to be non-predicate, 1) the second word is reset to next predicate candidate if the answer is non-predicate, otherwise, 2) the first word of the pair is reset to the predicate that is just determined, and the second is set to every argument candidates one by one. The classifier will scan the input sentence from left to right to check if each word is a true predicate.

Without any constraint, all word pairs in an input sequence must be considered by the classifier, leading to poor computational efficiency and unnecessary performance loss. Thus, the training sample for SRL task needs to be pruned properly.

We use a simple strategy to prune predicate candidates, namely, only verbs and nouns are chosen in this case.

There are two paths to collect argument candidates over the sequence. One is based on an input syntactic dependency tree, the other is based on a linear path of the sentence. As for the former (hereafter it is referred to *synPth*), we continue to use a dependency version of the pruning algorithm of (Xue and Palmer, 2004). The pruning algorithm is readdressed as the following.

> Initialization: Set the given predicate as the current node;
>
> (1) The current node and all of its syntactic children are selected as argument candidates (children are traversed from left to right.).
>
> (2) Reset the current node to its syntactic head and repeat step (1) until the root is reached.

Note that this pruning algorithm is slightly different from that of (Xue and Palmer, 2004), the predicate itself is also included in the argument candidate list as the nominal predicate sometimes takes itself as its argument.

The above pruning algorithm has been shown effective. However, it is still inefficient for a SRL

[1]CoNLL-2008 is an English-only task, while CoNLL-2009 is a multilingual one. Though the English corpus in CoNLL-2009 is almost identical to the corpus in the CoNLL-2008 shared task evaluation, the latter holds more sophisticated input structure as in (Surdeanu et al., 2008). The most difference for these two tasks is that the identification of semantic predicates is required in the task of CoNLL-2008 but not in CoNLL-2009.

system that needs to tackle argument identification/classification in a single stage. Assuming that arguments trend to surround their predicate, an assistant argument label '*_NoMoreArgument*' is introduced for further pruning. If an argument candidate in the above algorithm is assigned to such a label, then the pruning algorithm will end immediately. In training, this assistant label means no more samples will be generated for the current predicate, while in test, the decoder will not search arguments any more. It will be seen that this adaptive technique more effectively prunes argument candidates without missing more true arguments.

Along the linear path (hereafter referred to *linPth*), the classifier will search all words before and after the predicate. Similar to the pruning algorithm for *synPth*, we also introduce two assistant argument labels '*_noLeft*' and '*_noRight*' to adaptively prune words too far away from the predicate.

To show how assistant argument labels actually work, we give an example for $linPth$. Suppose an input sequence with argument labels for a predicate is

$a\ \ b\ \ c\ \ d\ \ \ e\ \ \ f\ \ \ g\ \ h\ \ .$

_ _ _ A1 _ A0 _ _ _

Note that c and g are two boundary words as no more arguments appear before or after them. After two assistant argument labels are added, it will be

$a\ \ b\ \ \ \ c\ \ \ \ \ d\ \ e\ \ f\ \ \ \ \ g\ \ \ \ \ h\ \ .$

_ _ *_noLeft* A1 _ A0 *_noRight* _ _

Training samples will generated from c to g according to the above sequence.

We use a Maximum Entropy classifier with a tunable Gaussian prior as usual. Our implementation of the model adopts L-BFGS algorithm for parameter optimization.

3 Feature Templates

3.1 Elements for Feature Generation

Motivated by previous works, we carefully consider those factors from a wide range of features that can help semantic role labeling for both predicate disambiguation, argument's identification and classification as the predicate is either verbal or nominal. These works include (Gildea and Jurafsky, 2002; Carreras and Marquez, 2005; Koomen et al., 2005; Marquez et al., 2005; Dang and Palmer, 2005; Pradhan et al., 2005; Toutanova et al., 2005; Jiang and Ng, 2006; Liu and Ng, 2007; Surdeanu et al., 2007; Johansson and Nugues, 2008; Che et al., 2008). Most feature templates that we will adopt for this work will come from various combinations or integrations of the following basic elements.

Word Property. This type of elements include word form (*form* and its split form, *spForm*)[2], lemma (*lemma*,*spLemma*), and part-of-speech tag (*pos*, *spPos*), syntactic dependency label (*dprel*), and semantic dependency label (*semdprel*)[3].

Syntactic Connection. This includes syntactic head (h), left(right) farthest(nearest) child (lm, ln, rm, and rn), and high(low) support verb or noun. We explain the last item, support verb(noun). From a given word to the syntactic root along the syntactic tree, the first verb/noun/preposition that is met is called as its low support verb/noun/preposition, and the nearest one to the root is called as its high support verb/noun/preposition. The concept of support verb was broadly used (Toutanova et al., 2005; Xue, 2006; Jiang and Ng, 2006)[4], we here extend it to nouns and prepositions. In addition, we introduce a slightly modified syntactic head, $pphead$, it returns the left most sibling of a given word if the word is headed by a preposition, otherwise it returns the original head.

Path. There are two basic types of path between the predicate and the argument candidates. One is the linear path (*linePath*) in the sequence, the other is the path in the syntactic parsing tree (*dpPath*). For the latter, we further divide it into four sub-types with respect to the syntactic root, *dpPath* is the full path in the syntactic tree. Leading two paths to the root from the predicate and the argument, respectively, the common part of these two paths will be *dpPathShare*. Assume that *dpPathShare* starts from a node r', then *dpPathPred* is from the predicate to r', and *dpPathArgu* is from the argument to r'.

Family. Two types of children sets for the predicate or argument candidate are considered, the

[2]In CoNLL-2008, Treebank tokens are split at the position that a hyphen (-) or a forward slash (/) occurs. This leads to two types of feature columns, non-split and split.

[3]Lemma and pos for either training or test are from automatically pre-analyzed columns in the input files.

[4]Note that the meaning of support verb is slightly different between (Toutanova et al., 2005) and (Xue, 2006; Jiang and Ng, 2006)

first includes all syntactic children (*children*), the second also includes all but excludes the left most and the right most children (*noFarChildren*).

Concatenation of Elements. For all collected elements according to *linePath*, *children* and so on, we use three strategies to concatenate all those strings to produce the feature value. The first is *seq*, which concatenates all collected strings without doing anything. The second is *bag*, which removes all duplicated strings and sort the rest. The third is *noDup*, which removes all duplicated neighbored strings.

We address some other elements that are not included by the above description as the following.

dpTreeRelation. It returns the relationship of a and p in the input syntactic tree. The possible values for this feature include `parent`, `sibling` etc.

isCurPred. It judges if a given word is the current predicate. If the word is the predicate, then it returns the predicate itself, otherwise it returns a default value.

existCross. It judges if a forthcoming dependency relation that is between a given word pair may cause any cross with all existing dependency relations.

distance. It counts the number of words along a given path, either $dpPath$ or $linePath$.

existSemdprel. It checks if the given argument label for other predicates has been assigned to a given word.

voice. This feature returns *Active* or *Passive* for verbs, and a default value for nouns.

baseline. Two types of semantic role baseline outputs are used for features from (Carreras and Marquez, 2005)[5]. $baseline\_Ax$ tags the head of the first NP before the predicate as $A0$ and the head of the first NP after the predicate as $A1$. $baseline\_Mod$ tags the dependant of the predicate as *AM-MOD* as it is a modal verb.

We show some feature template examples derived from the above mentioned items.

a.lm.lemma The lemma of the left most child of the argument candidate.

p.h.dprel The dependant label of the syntactic head of the predicate candidate.

p$_{-1}$*.pos+p.pos pos* of the previous word of the predicate and *PoS* of the predicate itself.

a:p|dpPath.lemma.bag Collect all lemmas along the syntactic tree path from the argument to the predicate, then removed all duplicated ones and sort the rest, finally concatenate all as a feature string.

a:p.highSupportNoun|linePath.dprel.seq Collect all dependant labels along with the line path from the argument to the high support noun of the predicate, then concatenate all as a feature string.

3.2 Feature Template Selection

Based on the above mentioned elements, 781 feature templates (hereafter the set of these templates is referred to FT)[6] are initially considered. Feature templates in this initial set are constructed in a generalized way. For example, if we find that a feature template *a.lm.lemma* was once used in some existing work, then such three templates, *a.rm.lemma*, *a.rn.lemma*, *a.ln.lemma* will be also added into the set.

As an optimal feature template subset cannot be expected to be extracted from so large a set by hand, a greedy feature selection similar to that in (Jiang and Ng, 2006; Ding and Chang, 2008) is applied. The detailed algorithm is described in Algorithm 1. Assuming that the number of feature templates in a given set is n, the algorithm of (Ding and Chang, 2008) requires $O(n^2)$ times of training/test routines, it cannot handle a set that consists of hundreds of templates. As the time complexity of Algorithm 1 is only $O(n)$, it permits a large scale feature selection accomplished by paying a reasonable time cost. Though the time complexity of the algorithm given by (Jiang and Ng, 2006) is also linear, it should assume all feature templates in the initial selected set 'good' enough and handles other feature template candidates in a strict incremental way. However, these two constraints are not easily satisfied in our case, while Algorithm 1 may release these two constraints.

Choosing the first 1/10 templates in FT as the initial selected set S, the feature selection is performed for two argument candidate traverse schemes, $synPth$ and $linPth$, respectively. 4686 machine learning routines run for the former, while 6248 routines for the latter. Two feature template sets, FT_{syn} and FT_{lin}, are obtained at last. These two sets are given in Table 1-3. We see that two sets share 30 identical feature templates as in Table 1. FT_{syn} holds 51 different templates

[5]These baseline rules were developed by Erik Tjong Kim Sang, from the University of Antwerp, Belgium.

[6]This set with detailed explanation will be available at our website.

_	*p.lm.dprel*
_	*p.rm.dprel*
_	*p.spForm*
_	p_{-1}*.spLemma*
_	*p.spLemma*
_	p_{-1}*.spLemma+p.spLemma*
_	*p.spLemma +* p_1*.spLemma*
_	*p.spLemma + p.h.spForm*
_	*p.spLemma + p.currentSense*
_	*p.lemma*
_	*p.lemma +* p_1*.lemma*
_	p_{-1}*.pos+p.pos*
_	*a.isCurPred.lemma*
_	a_{-2}*.isCurPred.lemma +* a_{-1}*.isCurPred.lemma*
_	*a.isCurPred.spLemma*
_	a_{-1}*.isCurPred.spLemma + a.isCurPred.spLemma*
_	*a.isCurPred.spLemma +* a_1*.isCurPred.spLemma*
_	*a.children.dprel.bag*
_	a_{-1}*.spLemma + a.spLemma*
_	a_{-1}*.spLemma + a.dprel*
_	a_{-1}*.spLemma + a.dprel + a.h.spLemma*
_	*a.lm*$_{-1}$*.spLemma*
_	*a.rm*$_{-1}$*.dprel + a.spPos*
_	a_{-1}*.lemma + a.dprel + a.h.lemma*
_	*a.lemma + p.lemma*
_	*a.pos + p.pos*
_	*a.spLemma + p.spLemma*
_	*a:p|dpPath.dprel*
_	*a:p|dpPathArgu.dprel*
_	*a:p|dpPathPred.spPos*

Table 1: Feature templates for both $synPth$ and $linPth$

as in Table 2 and FT_{lin} holds 57 different templates as in Table 3. In these tables, the subscripts -2(or -1) and 1(or 2) stand for the previous and next words, respectively. For example, *a.lm*$_{-1}$*.lemma* returns the lemma of the previous word of the argument's left most child.

4 Decoding

After the predicate sense is disambiguated, an optimal argument structure for each predicate is determined by the following maximal probability.

$$S_p = \text{argmax} \prod_i P(a_i|a_{i-1}, a_{i-2}, ...), \qquad (1)$$

where S_p is the argument structure, $P(a_i|a_{i-1}...)$ is the conditional probability to determine the label of the i-th argument candidate label. A beam search algorithm is used to find the optimal argument structure.

5 Evaluation Results

Our evaluation is performed on the standard training/development/test corpus of CoNLL-2008 shared task. The data is derived by merging a dependency version of the Penn Treebank with PropBank and NomBank. More details on the data are

Algorithm 1 Greedy Feature Selection

Input:
The set of all feature templates: FT
The set of selected feature templates: S_0

Output:
The set of selected feature templates: S

Procedure:

Let the counter $i = 1$
Let $S_i = S_0$ and $C = FT - S_i$
while do
  Train a model with features according to S_i, test on development set and the result is p_i.
  Let $C_r = null$.
  for each feature template f_j in set S_i **do**
    Let $S' = S_i - f_j$.
    Train a model with features according to S', test on development set and the result is p'.
    if $p' > p_i$ **then**
      $C_r = C_r + f_j$.
    end if
  end for
  $C = C + C_r$
  $S_i = S_i - C_r$
  Let $S'_i = S_i$
  Train a model with features according to S'_i, test on development set and the result is q_i.
  Let $C_r = null$
  for each feature template f_j in set C **do**
    Let $C' = S'_i + f_j$.
    Train a model with features according to C', test on development set and the result is p'.
    if $p' > q_i$ **then**
      $C_r = C_r + f_j$.
    end if
  end for
  $C = C - C_r$
  $S'_i = S'_i + C_r$
  if $S_i = S_{i-1}$(No feature templates are added or removed) or, neither p_i nor q_i is larger than p_{i-1} and q_{i-1} **then**
    Output $S = \text{argmax}_{p_i,q_i}\{S_i, S'_i\}$ and the algorithm ends.
  else
    Let $i = i + 1$, S_i=S_{i-1} and $C = FT - S_i$
  end if
end while

_	p_{-1}.lemma + p.lemma
_	p_{-2}.pos
_	p.pos
_	p_{-2}.spForm + p_{-1}.spForm
_	p_1.spForm
_	p.spForm + p.children.dprel.noDup
_	p.lm.spPos
_	p.spForm + p.lm.spPos + p.noFarChildren.spPos.bag + p.rm.spPos
_	p.dprel
_	p.children.dprel.bag
_	p.children.pos.seq
_	p.dprel = OBJ ? [a]
_	a.dprel
_	a_{-1}.lemma + a_1.lemma
_	a_1.lemma
_	a_{-1}.pos
_	a_1.spPos
_	a.h.lemma
_	a.h.spLemma
_	a.pphead.lemma
_	a.pphead.spLemma
_	a.lm.dprel + a.spPos
_	$a.rm_{-1}$.pos
_	a.spLemma + a.h.spPos
_	a.existSemdprel_A1
_	a.dprel = OBJ ?
_	a.form + a.children.pos.seq
_	a.children.adv.bag[b]
_	a:p|linePath.distance
_	a:p|dpPath.distance
_	a:p|existCross
_	a:p|dpPath.dprel.bag
_	a:p|dpPathPred.dprel.bag
_	a:p|dpPath.spForm.seq
_	a:p|dpPathArgu.spForm.seq
_	a:p|dpPathPred.spForm.bag
_	a:p|dpPath.spLemma.seq
_	a:p|dpPathArgu.spLemma.seq
_	a:p|dpPathArgu.spLemma.bag
_	a:p|dpPathPred.spLemma.bag
_	a:p|dpPath.spPos.bag
_	a:p|dpPathPred.spPos.bag
_	(a:p|dpPath.dprel.seq) + p.spPos
_	(a:p|dpTreeRelation) + a.spPos
_	(a:p|dpTreeRelation) + p.spPos
_	(a.highSupportVerb:p|dpTreeRelation) + a.spPos
_	a.highSupportNoun:p|dpPath.dprel.seq
_	a.lowSupportVerb:p|dpPath.dprel.seq
_	a:p|linePath.spForm.bag
_	a:p|linePath.spLemma.bag
_	a:p|linePath.spLemma.seq

[a] This feature checks if the dependant type is *OBJ*.
[b] *adv* means all adverbs.

Table 2: Feature templates only for *synPth*

_	p.currentSense + a.spLemma
_	p.currentSense + a.spPos
_	p.voice + (a:p|direction)
_	p.rm.dprel
_	p.children.dprel.noDup
_	p.rm.form
_	p.lowSupportNoun.spForm
_	p.lowSupportProp:p|dpTreeRelation
_	p_{-2}.form + p_{-1}.form
_	p.voice
_	p.form + p.children.dprel.noDup
_	p.pos + p.dprel
_	p.spForm + p.children.dprel.bag
_	a.voice + (a:p|direction)
_	a_{-1}.isCurPred.lemma
_	a_1.isCurPred.lemma
_	a_{-1}.isCurPred.lemma + a.isCurPred.lemma
_	a.isCurPred.lemma + a_1.isCurPred.lemma
_	a_1.isCurPred.spLemma
_	a_{-2}.isCurPred.spLemma + a_{-1}.isCurPred.spLemma
_	a.baseline_Ax + a.voice + (a:p|direction)
_	a.baseline_Mod
_	a.h.children.dprel.bag
_	a.lm.dprel + a.dprel
_	a.lm.dprel + a.pos
_	$a.lm_{-1}$.lemma
_	a.lm.lemma
_	$a.lm_1$.lemma
_	a.lm.pos + a.pos
_	a.lm.spForm
_	$a.lm_{-1}$.spPos
_	a.lm.spPos
_	a.ln.dprel + a.pos
_	a.noFarChildren.spPos.bag + a.rm.spPos
_	a.children.spPos.seq + p.children.spPos.seq
_	a.rm.dprel + a.pos
_	$a.rm_{-1}$.spPos
_	a.rm.spPos
_	$a.rm_1$.spPos
_	a.rn.dprel + a.spPos
_	a.form
_	a.form + a_1.form
_	a.form + a.pos
_	a_{-1}.lemma
_	a_{-1}.lemma + a.lemma
_	a_{-2}.pos
_	a.spForm + a_1.spForm
_	a.spForm + a.spPos
_	a.spLemma + a_1.spLemma
_	a.spForm + a.children.spPos.seq
_	a.spForm + a.children.spPos.bag
_	a.spLemma + a.h.spForm
_	a.spLemma + a.pphead.spForm
_	a.existSemdprel_A2
_	a:p|dpPathArgu.pos.seq
_	a:p|dpPathPred.dprel.seq
_	a:p|dpTreeRelation

Table 3: Feature templates only for *linPth*

in (Surdeanu et al., 2008). Note that CoNLL-2008 shared task is essentially a joint learning task for both syntactic and semantic dependencies, however, we will focus on semantic part of this task. The main semantic measure that we adopt is semantic labeled F_1 score (Sem-F_1). In addition, the macro labeled F_1 scores (Macro-F_1), which was used for the ranking of the participating systems of CoNLL-2008, the ratio between labeled F_1 score for semantic dependencies and the LAS for syntactic dependencies (Sem-F_1/LAS), are also given for reference.

5.1 Syntactic Dependency Parsers

We consider three types of syntactic information to feed the SRL task. One is gold-standard syntactic input, and other two are based on automatically parsing results of two parsers, the state-of-the-art syntactic parser described in (Johansson and Nugues, 2008)[7](it is referred to *Johansson*) and an integrated parser described as the following (referred to *MST*$_{ME}$).

The parser is basically based on the MSTParser[8] using all the features presented by (McDonald et al., 2006) with projective parsing. Moreover, we exploit three types of additional features to improve the parser. 1) Chen et al. (2008) used features derived from short dependency pairs based on large-scale auto-parsed data to enhance dependency parsing. Here, the same features are used, though all dependency pairs rather than short dependency pairs are extracted along with the dependency direction from training data rather than auto-parsed data. 2) Koo et al. (2008) presented new features based on word clusters obtained from large-scale unlabeled data and achieved large improvement for English and Czech. Here, the same features are also used as word clusters are generated only from the training data. 3) Nivre and McDonald (2008) presented an integrating method to provide additional information for graph-based and transition-based parsers. Here, we represent features based on dependency relations predicted by transition-based parsers for the MSTParer. For the sake of efficiency, we use a fast transition-based parser based on maximum entropy as in Zhao and Kit (2008). We still use the similar feature notations of that work.

[7]It is a 2-order maximum spanning tree parser with pseudo-projective techniques. A syntactic-semantic reranking was performed to output the final results according to (Johansson and Nugues, 2008). However, only 1-best outputs of the parser before reranking are used for our evaluation. Note that the reranking may slightly improve the syntactic performance according to (Johansson and Nugues, 2008).

[8]It's freely available at http://mstparser.sourceforge.net.

Parser	Path	*Adaptive Pruning*		Coverage
		/wo	/w	Rate
Gold	$synPth$	2.13M	1.05M (49.30%)	98.4%
	$linPth$	5.29M	1.57M (29.68%)	100.0%
Johansson	$synPth$	2.15M	1.06M (49.30%)	95.4%
	$linPth$	5.28M	1.57M (29.73%)	100.0%
MST_{ME}	$synPth$	2.15M	1.06M (49.30%)	95.0%
	$linPth$	5.29M	1.57M (29.68%)	100.0%

Table 4: The number of training samples on argument candidates

Syn-Parser	LAS	*synPth*+FT_{syn}		*linPth*+FT_{lin}	
		Sem F_1	Sem-F_1 /LAS	Sem F_1	Sem-F_1 /LAS
MST$_{ME}$	88.39	80.53	91.10	79.83	90.31
Johansson	89.28	80.94	90.66	79.84	89.43
Gold	100.00	84.57	84.57	83.34	83.34

Table 5: Semantic Labeled F_1

5.2 The Results

At first, we report the effectiveness of the proposed adaptive argument pruning. The numbers of argument candidates are in Table 4. The statistics is conducted on three different syntactic inputs. The coverage rate in the table means the ratio of how many true arguments are covered by the selected pruning scheme. Note that the adaptive pruning of argument candidates using assistant labels does not change this rate. This ratio only depends on which path, either $synPth$ or $linPth$, is chosen, and how good the syntactic input is (if $synPth$ is the case). From the results, we see that more than a half of argument candidates can be effectively pruned for $synPth$ and even 2/3 for $linPth$. As mentioned by (Pradhan et al., 2004), argument identification plays a bottleneck role in improving the performance of a SRL system. The effectiveness of the proposed additional pruning techniques may be seen as a significant improvement over the original algorithm of (Xue and Palmer, 2004). The results also indicate that such an assumption holds that arguments trend to close with their predicate, at either type of distance, syntactic or linear.

Based on different syntactic inputs, we obtain different results on semantic dependency parsing

as shown in Table 5. These results on different syntactic inputs also give us a chance to observe how semantic performance varies according to syntactic performance. The fact from the results is that the ratio Sem-F_1/LAS becomes relatively smaller as the syntactic input becomes better. Though not so surprised, the results do show that the argument traverse scheme $synPth$ always outperforms the other $linPth$. The result of this comparison partially shows that an integrated semantic role labeler is sensitive to the order of how argument candidates are traversed to some extent.

The performance given by $synPth$ is compared to some other systems that participated in the CoNLL-2008 shared task. They were chosen among the 20 participating systems either because they held better results (the first four participants) or because they used some joint learning techniques (Henderson et al., 2008). The results of (Titov et al., 2009) that use the similar joint learning technique as (Henderson et al., 2008) are also included[9]. Results of these evaluations on the test set are in Table 6. Top three systems of CoNLL-2008, (Johansson and Nugues, 2008; Ciaramita et al., 2008; Che et al., 2008), used SRL pipelines.

In this work, we partially use the similar techniques ($synPth$) for our participation in the shared tasks of CoNLL-2008 and 2009 (Zhao and Kit, 2008; Zhao et al., 2009b; Zhao et al., 2009a). Here we report that all SRL sub-tasks are tackled in one integrated model, while the predicate disambiguation sub-task was performed individually in both of our previous systems. Therefore, this is our first attempt at a full integrated SRL system.

(Titov et al., 2009) reported the best result by using joint learning technique up to now. The comparison indicates that our integrated system outputs a result quite close to the state-of-the-art by the pipeline system of (Johansson and Nugues, 2008) as the same syntactic structure input is adopted. It is worth noting that our system actually competes with two independent sub-systems of (Johansson and Nugues, 2008), one for verbal predicates, the other for nominal predicates. In addition, the results of our system is obtained without using additional joint learning technique like syntactic-semantic reranking. It indicates that our system is expected to obtain some further performance improvement by using such techniques.

[9]In addition, the work of (Henderson et al., 2008) and (Titov et al., 2009) jointly considered syntactic and semantic dependencies, that is significantly different from the others.

6 Conclusion

We have described a dependency-based semantic role labeling system for English from NomBank and PropBank. From the evaluations, the result of our system is quite close to the state of the art. As to our knowledge, it is the first integrated SRL system that achieves such a competitive performance against previous pipeline systems.

According to the path that the word-pair classifier traverses argument candidates, two integration schemes are presented. Argument candidate pruning and feature selection are performed on them, respectively. These two schemes are more than providing a trivial comparison. As assistant labeled are introduced to help further argument candidate pruning, and this techniques work well for both schemes, it support the assumption that arguments trend to surround their predicate. The proposed feature selection procedure also work for both schemes and output quite different two feature template sets, and either of the sets helps the system obtain a competitive performance, this fact suggests that the feature selection procedure is robust and effective, too.

Either of the presented integrated systems can provide a competitive performance. This conclusion about basic learning scheme for SRL is some different from previous literatures. However, according to our results, there does exist a 'harmony' feature template set that is helpful to both predicate and argument identification/classification, or SRL for both verbal and nominal predicates. We attribute this different conclusion to two main factors, 1) much more feature templates (for example, ten times more than those used by Xue et al.) than previous that are considered for a successful feature engineering, 2) a maximum entropy classifier makes it possible to accept so many various features in one model. Note that maximum entropy is not so sensitive to those (partially) overlapped features, while SVM and other margin-based learners are not so.

Acknowledgements

Our thanks give to Dr. Richard Johansson, who kindly provided the syntactic output for his participation in the CoNLL-2008 shared task.

Systems[a]	LAS	Sem-F_1	Macro F_1	Sem-F_1 /LAS	pred-F_1[b]	argu-F_1[c]	Verb-F_1[d]	Nomi-F_1[e]
Johansson:2008*[f]	**89.32**	**81.65**	**85.49**	**91.41**	**87.22**	**79.04**	**84.78**	77.12
Ours:*Johansson*	89.28	80.94	85.12	90.66	86.57	78.30	83.66	76.93
Ours:MST_{ME}	88.39	80.53	84.93	91.10	86.80	77.60	82.77	**77.23**
Johansson:2008	**89.32**	80.37	84.86	89.98	85.40	78.02	84.45	74.32
Ciaramita:2008*	87.37	78.00	82.69	89.28	83.46	75.35	80.93	73.80
Che:2008	86.75	78.52	82.66	90.51	85.31	75.27	80.46	75.18
Zhao:2008*	87.68	76.75	82.24	87.53	78.52	75.93	78.81	73.59
Ciaramita:2008	86.60	77.50	82.06	89.49	83.46	74.56	80.15	73.17
Titov:2009	87.50	76.10	81.80	86.97	–	–	–	–
Zhao:2008	86.66	76.16	81.44	87.88	78.26	75.18	77.67	73.28
Henderson:2008*	87.64	73.09	80.48	83.40	81.42	69.10	75.84	68.90
Henderson:2008	86.91	70.97	79.11	81.66	79.60	66.83	73.80	66.26
Ours:*Gold*	100.0	84.57	92.20	84.57	87.67	83.15	88.71	78.39

[a]Ranking according to Sem-F_1

[b]Labeled F_1 for predicate identification and classification

[c]Labeled F_1 for argument identification and classification

[d]Labeled F_1 for verbal predicates

[e]Labeled F_1 for nominal predicates

[f]* means post-evaluation results, which are available at the official website of CoNLL-2008 shared task, http://www.yr-bcn.es/dokuwiki/doku.php?id=conll2008:start.

Table 6: Comparison of the best existing systems

References

Xavier Carreras and Lluis Marquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL-2005*, pages 152–164, Ann Arbor, Michigan, USA.

Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu, and Sheng Li. 2008. A cascaded syntactic and semantic dependency parsing system. In *Proceedings of CoNLL-2008*, pages 238–242, Manchester, England, August.

Wenliang Chen, Daisuke Kawahara, Kiyotaka Uchimoto, Yujie Zhang, and Hitoshi Isahara. 2008. Dependency parsing with short dependency relations in unlabeled data. In *Proceedings of IJCNLP-2008*, Hyderabad, India, January 8-10.

Massimiliano Ciaramita, Giuseppe Attardi, Felice Dell'Orletta, and Mihai Surdeanu. 2008. Desrl: A linear-time semantic role labeling system. In *Proceedings of CoNLL-2008*, pages 258–262, Manchester, England, August.

Hoa Trang Dang and Martha Palmer. 2005. The role of semantic roles in disambiguating verb senses. In *Proceedings of ACL-2005*, pages 42–49, Ann Arbor, USA.

Weiwei Ding and Baobao Chang. 2008. Improving chinese semantic role classification with hierarchical feature selection strategy. In *Proceedings of EMNLP-2008*, pages 324–323, Honolulu, USA.

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

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*, pages 1–18, Boulder, Colorado, USA.

James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proceedings of CoNLL-2008*, pages 178–182, Manchester, England, August.

Zheng Ping Jiang and Hwee Tou Ng. 2006. Semantic role labeling of nombank: A maximum entropy approach. In *Proceedings of EMNLP-2006*, pages 138–145, Sydney, Australia.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic – semantic analysis with propbank and nombank. In *Proceedings of CoNLL-2008*, page 183 – 187, Manchester, UK.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, USA, June.

Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of CoNLL-2005*, pages 181–184, Ann Arbor, Michigan, USA.

Chang Liu and Hwee Tou Ng. 2007. Learning predictive structures for semantic role labeling of nombank. In *Proceedings of ACL-2007*, pages 208–215, Prague, Czech.

Lluis Marquez, Mihai Surdeanu, Pere Comas, and Jordi Turmo. 2005. A robust combination strategy for semantic role labeling. In *Proceedings of HLT/EMNLP-2005*, page 644 ‑ 651, Vancouver, Canada.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL-X*, New York City, June.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The nombank project: An interim report. In *Proceedings of HLT/NAACL Workshop on Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA, May 6.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio, June.

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

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of HLT/NAACL-2004*, pages 233–240, Boston, Massachusetts, USA.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of ACL-2005*, pages 581–588, Ann Arbor, USA.

James Pustejovsky, Adam Meyers, Martha Palmer, and Massimo Poesio. 2005. Merging propbank, nombank, timebank, penn discourse treebank and coreference. In *Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky*, pages 5–12, Ann Arbor, USA.

Mihai Surdeanu, Lluis Marquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL-2008*, pages 159–177, Manchester, UK.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *IJCAI-2009*, Pasadena, California, USA.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL-2005*, pages 589–596, Ann Arbor, USA.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP-2004*, pages 88–94, Barcelona, Spain, July 25-26.

Nianwen Xue. 2006. Semantic role labeling of nominalized predicates in chinese. In *Proceedings of NAACL-2006*, pages 431–438, New York City, USA, June.

Hai Zhao and Chunyu Kit. 2008. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *Proceeding of CoNLL-2008*, pages 203–207, Manchester, UK.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009a. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*, pages 61–66, Boulder, Colorado, USA.

Hai Zhao, Wenliang Chen, Chunyu Kit, and Guodong Zhou. 2009b. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of CoNLL-2009*, pages 55–60, Boulder, Colorado, USA.

First- and Second-Order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests*

Zhifei Li and **Jason Eisner**
Department of Computer Science and Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218, USA
zhifei.work@gmail.com, jason@cs.jhu.edu

Abstract

Many statistical translation models can be regarded as weighted logical deduction. Under this paradigm, we use weights from the expectation semiring (Eisner, 2002), to compute first-order statistics (e.g., the expected hypothesis length or feature counts) over packed forests of translations (lattices or hypergraphs). We then introduce a novel *second-order* expectation semiring, which computes second-order statistics (e.g., the variance of the hypothesis length or the gradient of entropy). This second-order semiring is essential for many interesting training paradigms such as minimum risk, deterministic annealing, active learning, and semi-supervised learning, where gradient descent optimization requires computing the gradient of entropy or risk. We use these semirings in an open-source machine translation toolkit, **Joshua**, enabling minimum-risk training for a benefit of up to 1.0 BLEU point.

1 Introduction

A hypergraph or "packed forest" (Gallo et al., 1993; Klein and Manning, 2004; Huang and Chiang, 2005) is a compact data structure that uses structure-sharing to represent exponentially many trees in polynomial space. A *weighted* hypergraph also defines a probability or other weight for each tree, and can be used to represent the hypothesis space considered (for a given input) by a monolingual parser or a tree-based translation system, e.g., tree to string (Quirk et al., 2005; Liu et al., 2006), string to tree (Galley et al., 2006), tree to tree (Eisner, 2003), or string to string with latent tree structures (Chiang, 2007).

*This research was partially supported by the Defense Advanced Research Projects Agency's GALE program via Contract No HR0011-06-2-0001. We are grateful to Sanjeev Khudanpur for early guidance and regular discussions.

Given a hypergraph, we are often interested in computing some quantities over it using dynamic programming algorithms. For example, we may want to run the Viterbi algorithm to find the most probable derivation tree in the hypergraph, or the k most probable trees. Semiring-weighted logic programming is a general framework to specify these algorithms (Pereira and Warren, 1983; Shieber et al., 1994; Goodman, 1999; Eisner et al., 2005; Lopez, 2009). Goodman (1999) describes many useful semirings (e.g., Viterbi, inside, and Viterbi-n-best). While most of these semirings are used in "testing" (i.e., decoding), we are mainly interested in the semirings that are useful for "training" (i.e., parameter estimation). The expectation semiring (Eisner, 2002), originally proposed for finite-state machines, is one such "training" semiring, and can be used to compute feature expectations for the E-step of the EM algorithm, or gradients of the likelihood function for gradient descent.

In this paper, we apply the expectation semiring (Eisner, 2002) to a hypergraph (or packed forest) rather than just a lattice. We then propose a novel *second-order* expectation semiring, nicknamed the "variance semiring."

The original first-order expectation semiring allows us to efficiently compute a vector of first-order statistics (expectations; first derivatives) on the set of paths in a lattice or the set of trees in a hypergraph. The second-order expectation semiring *additionally* computes a matrix of second-order statistics (expectations of *products*; second derivatives (Hessian); derivatives of expectations).

We present details on how to compute many interesting quantities over the hypergraph using the expectation and variance semirings. These quantities include expected hypothesis length, feature expectation, entropy, cross-entropy, Kullback-Leibler divergence, Bayes risk, variance of hypothesis length, gradient of entropy and Bayes risk, covariance and Hessian matrix, and so on. The variance semiring is essential for many interesting training paradigms such as deterministic

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 40–51,
Singapore, 6-7 August 2009.

annealing (Rose, 1998), minimum risk (Smith and Eisner, 2006), active and semi-supervised learning (Grandvalet and Bengio, 2004; Jiao et al., 2006). In these settings, we must compute the gradient of entropy or risk. The semirings can also be used for second-order gradient optimization algorithms.

We implement the expectation and variance semirings in **Joshua** (Li et al., 2009a), and demonstrate their practical benefit by using minimum-risk training to improve Hiero (Chiang, 2007).

2 Semiring Parsing on Hypergraphs

We use a specific tree-based system called Hiero (Chiang, 2007) as an example, although the discussion is general for any systems that use a hypergraph to represent the hypothesis space.

2.1 Hierarchical Machine Translation

In Hiero, a synchronous context-free grammar (SCFG) is extracted from automatically word-aligned corpora. An illustrative grammar rule for Chinese-to-English translation is

$$X \;\rightarrow\; \langle\, X_0 \text{ 的 } X_1 \,,\, X_1 \text{ of } X_0 \,\rangle\,,$$

where the Chinese word 的 means *of*, and the alignment, encoded via subscripts on the nonterminals, causes the two phrases around 的 to be reordered around *of* in the translation. Given a source sentence, Hiero uses a CKY parser to generate a hypergraph, encoding many derivation trees along with the translation strings.

2.2 Hypergraphs

Formally, a hypergraph is a pair $\langle V, E\rangle$, where V is a set of *nodes* (vertices) and E is a set of *hyperedges*, with each hyperedge connecting a *set* of *antecedent nodes* to a single *consequent node*.[1] In parsing parlance, a node corresponds to an *item* in the chart (which specifies aligned spans of input and output together with a nonterminal label). The root node corresponds to the *goal item*. A hyperedge represents an SCFG rule that has been "instantiated" at a particular position, so that the nonterminals on the right and left sides have been replaced by particular antecedent and consequent items; this corresponds to storage of backpointers in the chart.

We write $T(e)$ to denote the set of antecedent nodes of a hyperedge e. We write $I(v)$ for the set of *incoming hyperedges* of node v (i.e., hyperedges of which v is the consequent), which represent different ways of deriving v. Figure 1 shows a simple Hiero-style hypergraph. The hypergraph encodes four different derivation trees that share some of the same items. By exploiting this sharing, a hypergraph can compactly represent exponentially many trees.

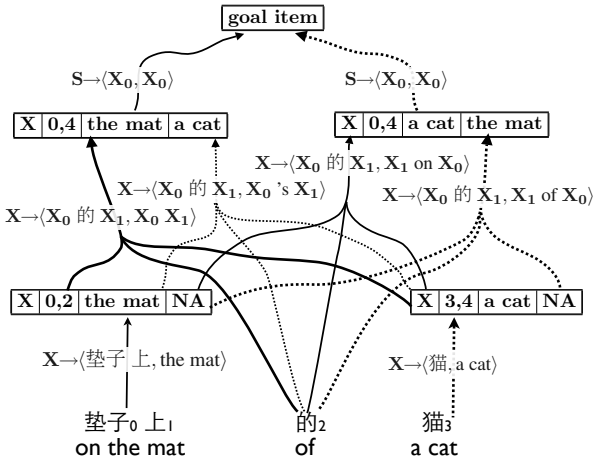


Figure 1: A toy hypergraph in Hiero. When generating the hypergraph, a trigram language model is integrated. Rectangles represent items, where each item is identified by the non-terminal symbol, source span, and left- and right-side language model states. An item has one or more incoming hyperedges. A hyperedge consists of a rule, and a pointer to an antecedent item for each non-terminal symbol in the rule.

We observe that any finite-state automaton can also be encoded as a hypergraph (in which every hyperedge is an ordinary edge that connects a *single* antecedent to a consequent). Thus, the methods of this paper apply directly to the simpler case of hypothesis lattices as well.

2.3 Semiring Parsing

We assume a hypergraph HG, which compactly encodes many derivation trees $d \in \mathrm{D}$. Given HG, we wish to extract the best derivations—or other aggregate properties of the forest of derivations. Semiring parsing (Goodman, 1999) is a general framework to describe such algorithms. To define a particular algorithm, we choose a semiring K and specify a "weight" $k_e \in K$ for each hyperedge e. The desired aggregate result then emerges as the *total weight* of all derivations in the hypergraph. For example, to simply count derivations, one can assign every hyperedge weight 1 in the semiring of *ordinary integers*; then each derivation also has weight 1, and their total weight is the number of derivations.

We write $\mathbf{K} = \langle K, \oplus, \otimes, \mathbf{0}, \mathbf{1}\rangle$ for a semiring with elements K, additive operation $\oplus$, multi-

[1] Strictly speaking, making each hyperedge designate a single consequent defines a *B-hypergraph* (Gallo et al., 1993).

plicative operation $\otimes$, additive identity $\mathbf{0}$, and multiplicative identity $\mathbf{1}$. The $\otimes$ operation is used to obtain the weight of each derivation d by *multiplying* the weights of its component hyperedges e, that is, $k_d = \bigotimes_{e \in d} k_e$. The $\oplus$ operation is used to *sum* over all derivations d in the hypergraph to obtain the *total* weight of the hypergraph HG, which is $\bigoplus_{d \in \mathrm{D}} \bigotimes_{e \in d} k_e$.[2] Figure 2 shows how to compute the total weight of an acyclic hypergraph HG.[3] In general, the total weight is a sum over exponentially many derivations d. But Figure 2 sums over these derivations in time only linear on the size of the hypergraph. Its correctness relies on axiomatic properties of the semiring: namely, $\oplus$ is associative and commutative with identity $\mathbf{0}$, $\otimes$ is associative with two-sided identity $\mathbf{1}$, and $\otimes$ distributes over $\oplus$ from both sides. The distributive property is what makes Figure 2 work. The other properties are necessary to ensure that $\bigoplus_{d \in \mathrm{D}} \bigotimes_{e \in d} k_e$ is well-defined.[4]

The algorithm in Figure 2 is general and can be applied with any semiring (e.g., Viterbi). Below, we present our novel semirings.

3 Finding Expectations on Hypergraphs

We now introduce the computational problems of this paper and the semirings we use to solve them.

3.1 Problem Definitions

We are given a function $p : \mathrm{D} \to \mathbb{R}_{\geq 0}$, which decomposes **multiplicatively** over component hyperedges e of a derivation $d \in \mathrm{D}$: that is, $p(d) \stackrel{\text{def}}{=} \prod_{e \in d} p_e$. In practice, $p(d)$ will specify a probability distribution over the derivations in the hypergraph. It is often convenient to permit this probability distribution to be unnormalized, i.e., one may have to divide it through by some Z to get a proper distribution that sums to 1.

We are also given two functions of interest $r, s : \mathrm{D} \to \mathbb{R}$, each of which decomposes **additively** over its component hyperedges e: that is, $r(d) \stackrel{\text{def}}{=} \sum_{e \in d} r_e$, and $s(d) \stackrel{\text{def}}{=} \sum_{e \in d} s_e$.

We are now interested in computing the following quantities on the hypergraph HG:

$$Z \stackrel{\text{def}}{=} \sum_{d \in \mathrm{D}} p(d) \quad (1)$$

$$\overline{r} \stackrel{\text{def}}{=} \sum_{d \in \mathrm{D}} p(d) r(d) \quad (2)$$

$$\overline{s} \stackrel{\text{def}}{=} \sum_{d \in \mathrm{D}} p(d) s(d) \quad (3)$$

$$\overline{t} \stackrel{\text{def}}{=} \sum_{d \in \mathrm{D}} p(d) r(d) s(d) \quad (4)$$

Note that $\overline{r}/Z$, $\overline{s}/Z$, and $\overline{t}/Z$ are expectations under p of $r(d)$, $s(d)$, and $r(d)s(d)$, respectively.

More formally, the probabilistic interpretation is that D is a discrete sample space (consisting

INSIDE(HG, $\mathbf{K}$)

```
for v in topological order on HG                ▷ each node
    ▷ find β(v) ← ⊕_{e∈I(v)} (k_e ⊗ (⊗_{u∈T(e)} β(u)))
    β(v) ← 0
    for e ∈ I(v)                                ▷ each incoming hyperedge
        k ← k_e                                 ▷ hyperedge weight
        for u ∈ T(e)                            ▷ each antecedent node
            k ← k ⊗ β(u)
    β(v) ← β(v) ⊕ k
return β(root)
```

Figure 2: Inside algorithm for an acyclic hypergraph HG, which provides hyperedge weights $k_e \in \mathbf{K}$. This computes all "inside weights" $\beta(v) \in \mathbf{K}$, and returns $\beta(\text{root})$, which is total weight of the hypergraph, i.e., $\bigoplus_{d \in \mathrm{D}} \bigotimes_{e \in d} k_e$.

OUTSIDE(HG, $\mathbf{K}$)

```
for v in HG
    α(v) ← 0
α(root) ← 1
for v in reverse topological order on HG
    for e ∈ I(v)                                ▷ each incoming hyperedge
        for u ∈ T(e)                            ▷ each antecedent node
            α(u) ← α(u) ⊕ (α(v) ⊗ k_e ⊗
                           ⊗_{w∈T(e),w≠u} β(w))
```

Figure 3: Computes the "outside weights" $\alpha(v)$. Can only be run after INSIDE(HG) of Figure 2 has already computed the inside weights $\beta(v)$.

[2] Eisner (2002) uses *closed semirings* that are also equipped with a Kleene closure operator *. For example, in the real semiring $\langle \mathbb{R}, +, \times, 0, 1 \rangle$, we define $p^* = (1-p)^{-1}$ $(= 1 + p + p^2 + \ldots)$ for $|p| < 1$ and is undefined otherwise. The closure operator enables exact summation over the *infinitely* many paths in a cyclic FSM, or trees in a hypergraph with non-branching cycles, without the need to iterate around cycles to numerical convergence. For completeness, we specify the closure operator for our semirings, satisfying the axioms $k^* = 1 \oplus k \otimes k^* = 1 \oplus k^* \otimes k$, but we do not use it in our experiments since our hypergraphs are acyclic.

[3] We assume that HG has *already* been built by deductive inference (Shieber et al., 1994). But in practice, the nodes' inside weights $\beta(v)$ are usually accumulated *as the hypergraph is being built*, so that pruning heuristics can consult them.

[4] Actually, the notation $\bigotimes_{e \in d} k_e$ assumes that $\otimes$ is commutative as well, as does the notation "for $u \in T(e)$" in our algorithms; neither specifies a loop order. One could however use a non-commutative semiring by ordering each hyperedge's antecedents and specifying that a derivation's weight is the product of the weights of its hyperedges *when visited in prefix order*. Tables 1–2 will not assume any commutativity.

INSIDE-OUTSIDE(HG, $\mathbf{K}$, X)

1 ▷ Run inside and outside on HG with only k_e weights
2 $\hat{k} \leftarrow$ INSIDE(HG, $\mathbf{K}$) ▷ see Figure 2
3 OUTSIDE(HG, $\mathbf{K}$) ▷ see Figure 3
4 ▷ Do a single linear combination to get $\hat{x}$
5 $\hat{x} \leftarrow \mathbf{0}$
6 **for** v **in** HG ▷ each node
7 **for** $e \in I(v)$ ▷ each incoming hyperedge
8 $\overline{k_e} \leftarrow \alpha(v)$
9 **for** $u \in T(e)$ ▷ each antecedent node
10 $\overline{k_e} \leftarrow \overline{k_e}\,\beta(u)$
11 $\hat{x} \leftarrow \hat{x} + (\overline{k_e}\, x_e)$
12 **return** $\langle \hat{k}, \hat{x} \rangle$

Figure 4: If every hyperedge specifies a weight $\langle k_e, x_e \rangle$ in some expectation semiring $\mathbb{E}_{\mathbf{K},X}$, then this inside-outside algorithm is a more efficient alternative to Figure 2 for computing the total weight $\langle \hat{k}, \hat{x} \rangle$ of the hypergraph, especially if the x_e are vectors. First, at lines 2–3, the inside and outside algorithms are run using only the k_e weights, obtaining only $\hat{k}$ (without $\hat{x}$) but also obtaining all inside and outside weights $\beta, \alpha \in \mathbf{K}$ as a side effect. Then the second component $\hat{x}$ of the total weight is accumulated in lines 5–11 as a linear combination of all the x_e values, namely $\hat{x} = \sum_e \overline{k_e} x_e$, where $\overline{k_e}$ is computed at lines 8–10 using α and β weights. The linear coefficient $\overline{k_e}$ is the "exclusive weight" for hyperedge e, meaning that the *product* $\overline{k_e} k_e$ is the total weight in $\mathbf{K}$ of all derivations $d \in$ D that include e.

of all derivations in the hypergraph), p is a measure over this space, and $r, s : \mathrm{D} \rightarrow \mathbb{R}$ are random variables. Then $\overline{r}/Z$ and $\overline{s}/Z$ give the expectations of these random variables, and $\overline{t}/Z$ gives the expectation of their product $t = rs$, so that $\overline{t}/Z - (\overline{r}/Z)(\overline{s}/Z)$ gives their covariance.

Example 1: $r(d)$ is the length of the translation corresponding to derivation d (arranged by setting r_e to the number of target-side terminal words in the SCFG rule associated with e). Then $\overline{r}/Z$ is the expected hypothesis length. ***Example 2:*** $r(d)$ evaluates the loss of d compared to a reference translation, using some additively decomposable loss function. Then $\overline{r}/Z$ is the risk (expected loss), which is useful in minimum-risk training. ***Example 3:*** $r(d)$ is the number of times that a certain feature fires on d. Then $\overline{r}/Z$ is the expected feature count, which is useful in maximum-likelihood training. We will generalize later in Section 4 to allow $r(d)$ to be a vector of features. ***Example 4:*** Suppose $r(d)$ and $s(d)$ are identical and both compute hypothesis length. Then the second-order statistic $\overline{t}/Z$ is the second moment of the length distribution, so the variance of hypothesis length can be found as $\overline{t}/Z - (\overline{r}/Z)^2$.

3.2 Computing the Quantities

We will use the semiring parsing framework to compute the quantities (1)–(4). Although each is a sum over exponentially many derivations, we will compute it in $O(|\mathrm{HG}|)$ time using Figure 2.

In the simplest case, let $\mathbf{K} = \langle \mathbb{R}, +, \times, 0, 1 \rangle$, and define $k_e = p_e$ for each hyperedge e. Then the algorithm of Figure 2 reduces to the classical inside algorithm (Baker, 1979) and computes Z.

Next suppose $\mathbf{K}$ is the expectation semiring (Eisner, 2002), shown in Table 1. Define $k_e = \langle p_e, p_e r_e \rangle$. Then Figure 2 will return $\langle Z, \overline{r} \rangle$.

Finally, suppose $\mathbf{K}$ is our novel *second-order* expectation semiring, which we introduce in Table 2. Define $k_e = \langle p_e, p_e r_e, p_e s_e, p_e r_e s_e \rangle$. Then the algorithm of Figure 2 returns $\langle Z, \overline{r}, \overline{s}, \overline{t} \rangle$. Note that, to compute $\overline{t}$, one cannot simply construct a *first-order* expectation semiring by defining $t(d) \stackrel{\text{def}}{=} r(d)s(d)$ because $t(d)$, unlike $r(d)$ and $s(d)$, is not additively decomposable over the hyperedges in d.[5] Also, when $r(d)$ and $s(d)$ are identical, the second-order expectation semiring allows us to compute variance as $\overline{t}/Z - (\overline{r}/Z)^2$, which is why we may call our second-order expectation semiring the **variance semiring**.

3.3 Correctness of the Algorithms

To prove our claim about the first-order expectation semiring, we first observe that the definitions in Table 1 satisfy the semiring axioms. The reader can easily check these axioms (as well as the closure axioms in footnote 2). With a valid semiring, we then simply observe that Figure 2 returns the total weight $\bigoplus_{d \in \mathrm{D}} \bigotimes_{e \in d} k_e = \bigoplus_{d \in \mathrm{D}} \langle p(d), p(d)r(d) \rangle = \langle Z, \overline{r} \rangle$. It is easy to verify the second equality from the definitions of $\oplus$, Z, and $\overline{r}$. The first equality requires proving that $\bigotimes_{e \in d} k_e = \langle p(d), p(d)r(d) \rangle$ from the definitions of $\otimes, k_e$, $p(d)$, and $r(d)$. The main intuition is that $\otimes$ can be used to build up $\langle p(d), p(d)r(d) \rangle$ inductively from the k_e: if d decomposes into two disjoint subderivations d_1, d_2, then $\langle p(d), p(d)r(d) \rangle = \langle p(d_1)p(d_2), p(d_1)p(d_2)(r(d_1) + r(d_2)) \rangle = \langle p(d_1), p(d_1)r(d_1) \rangle \otimes \langle p(d_2), p(d_2)r(d_2) \rangle$. The base cases are where d is a single hyperedge e, in which case $\langle p(d), p(d)r(d) \rangle = k_e$ (thanks to our choice of k_e), and where d is empty, in which case

[5] However, in a more tricky way, the second-order expectation semiring can be constructed using the first-order expectation semiring, as will be seen in Section 4.3.

Element	$\langle p, r \rangle$
$\langle p_1, r_1 \rangle \otimes \langle p_2, r_2 \rangle$	$\langle p_1 p_2,\ p_1 r_2 + p_2 r_1 \rangle$
$\langle p_1, r_1 \rangle \oplus \langle p_2, r_2 \rangle$	$\langle p_1 + p_2,\ r_1 + r_2 \rangle$
$\langle p, r \rangle^*$	$\langle p^*, p^* p^* r \rangle$
$\mathbf{0}$	$\langle 0, 0 \rangle$
$\mathbf{1}$	$\langle 1, 0 \rangle$

Table 1: **Expectation semiring**: Each element in the semiring is a **pair** $\langle p, r \rangle$. The second and third rows define the operations between two elements $\langle p_1, r_1 \rangle$ and $\langle p_2, r_2 \rangle$, and the last two rows define the identities. Note that the multiplicative identity $\mathbf{1}$ has an r component of 0.

s_a	s_b	$a + b$		$a \cdot b$	
		s_{a+b}	ℓ_{a+b}	$s_{a \cdot b}$	$\ell_{a \cdot b}$
+	+	+	$\ell_a + \log(1 + e^{\ell_b - \ell_a})$	+	$\ell_a + \ell_b$
+	-	+	$\ell_a + \log(1 - e^{\ell_b - \ell_a})$	-	$\ell_a + \ell_b$
-	+	-	$\ell_a + \log(1 - e^{\ell_b - \ell_a})$	-	$\ell_a + \ell_b$
-	-	-	$\ell_a + \log(1 + e^{\ell_b - \ell_a})$	+	$\ell_a + \ell_b$

Table 3: **Storing signed values in log domain**: each value a $(= s_a e^{\ell_a})$ is stored as a **pair** $\langle s_a, \ell_a \rangle$ where s_a and ℓ_a are the **sign bit** of a and **natural logarithm** of $|a|$, respectively. This table shows the operations between two values $a = s_a 2^{\ell_a}$ and $b = s_b 2^{\ell_b}$, assuming $\ell_a \geq \ell_b$. *Note:* $\log(1 + x)$ (where $|x| < 1$) should be computed by the Mercator series $x - x^2/2 + x^3/3 - \cdots$, e.g., using the math library function log1p.

$\langle p(d), p(d) r(d) \rangle = \mathbf{1}$. It follows by induction that $\langle p(d), p(d) r(d) \rangle = \bigotimes_{e \in d} k_e$.

The proof for the second-order expectation semiring is similar. In particular, one mainly needs to show that $\bigotimes_{e \in d} k_e = \langle p(d), p(d) r(d), p(d) s(d), p(d) r(d) s(d) \rangle$.

3.4 Preventing Underflow/Overflow

In Tables 1–2, we do not discuss how to store p, r, s, and t. If p is a probability, it often suffers from the underflow problem. r, s, and t may suffer from both underflow and overflow problems, depending on their scales.

To address these, we could represent p in the log domain as usual. However, r, s, and t can be positive or negative, and we cannot directly take the log of a negative number. Therefore, we represent real numbers as ordered pairs. Specifically, to represent $a = s_a e^{\ell_a}$, we store $\langle s_a, \ell_a \rangle$, where the $s_a \in \{+, -\}$ is the **sign bit** of a and the floating-point number ℓ_a is the **natural logarithm** of $|a|$.[6] Table 3 shows the "$\cdot$" and "$+$"operations.

[6] An alternative that avoids log and exp is to store $a = f_a 2^{e_a}$ as $\langle f_a, e_a \rangle$, where f_a is a *floating-point number* and e_a is a sufficiently wide *integer*. E.g., combining a 32-bit f_a with a 32-bit e_a will in effect extend f_a's 8-bit internal exponent to 32 bits by adding e_a to it. This gives much more dynamic range than the 11-bit exponent of a 64-bit double-precision floating-point number, if vastly less than in Table 3.

4 Generalizations and Speedups

In this section, we generalize beyond the above case where p, r, s are $\mathbb{R}$-valued. In general, p may be an element of some other semiring, and r and s may be **vectors** or other algebraic objects.

When r and s are vectors, especially high-dimensional vectors, the basic "inside algorithm" of Figure 2 will be slow. We will show how to speed it up with an "inside-outside algorithm."

4.1 Allowing Feature Vectors and More

In general, for P, R, S, T, we can define the first-order expectation semiring $\mathbb{E}_{P,R} = \langle P \times R, \oplus, \otimes, \mathbf{0}, \mathbf{1} \rangle$ and the second-order expectation semiring $\mathbb{E}_{P,R,S,T} = \langle P \times R \times S \times T, \oplus, \otimes, \mathbf{0}, \mathbf{1} \rangle$, using the definitions from Tables 1–2. But do those definitions remain meaningful, and do they continue to satisfy the semiring axioms?

Indeed they do when $P = \mathbb{R}, R = \mathbb{R}^n, S = \mathbb{R}^m, T = \mathbb{R}^{n \times m}$, with rs defined as the outer product $rs^{\mathbf{T}}$ (a matrix) where $s^{\mathbf{T}}$ is the transpose of s. In this way, the second-order semiring $\mathbb{E}_{P,R,S,T}$ lets us take expectations of vectors and outer products of vectors. So we can find *means* and *covariances* of any number of linearly decomposable quantities (e.g., feature counts) defined on the hypergraph.

We will consider some other choices in Sections 4.3–4.4 below. Thus, for generality, we conclude this section by stating the precise technical conditions needed to construct $\mathbb{E}_{P,R}$ and $\mathbb{E}_{P,R,S,T}$:

- P is a semiring
- R is a P-module (e.g, a vector space), meaning that it comes equipped with an associative and commutative addition operation with an identity element 0, and also a multiplication operation $P \times R \to R$, such that $p(r_1 + r_2) = pr_1 + pr_2$, $(p_1 + p_2)r = p_1 r + p_2 r$, $p_1(p_2 r) = (p_1 p_2)r$
- S and T are also P-modules
- there is a multiplication operation $R \times S \to T$ that is bilinear, i.e., $(r_1 + r_2)s = r_1 s + r_2 s$, $r(s_1 + s_2) = rs_1 + rs_2$, $(pr)s = p(rs)$, $r(ps) = p(rs)$

As a matter of notation, note that above and in Tables 1–2, we overload "+" to denote any of the addition operations within P, R, S, T; overload "0" to denote their respective additive identities; and overload concatenation to denote any of the multiplication operations within or between

Element	$\langle p, r, s, t \rangle$
$\langle p_1, r_1, s_1, t_1 \rangle \otimes \langle p_2, r_2, s_2, t_2 \rangle$	$\langle p_1 p_2,\ p_1 r_2 + p_2 r_1,\ p_1 s_2 + p_2 s_1,$ $p_1 t_2 + p_2 t_1 + r_1 s_2 + r_2 s_1 \rangle$
$\langle p_1, r_1, s_1, t_1 \rangle \oplus \langle p_2, r_2, s_2, t_2 \rangle$	$\langle p_1 + p_2,\ r_1 + r_2,\ s_1 + s_2,\ t_1 + t_2 \rangle$
$\langle p, r, s, t \rangle^*$	$\langle p^*,\ p^* p^* r,\ p^* p^* s,\ p^* p^* (p^* r s + p^* r s + t) \rangle$
$\mathbf{0}$	$\langle 0, 0, 0, 0 \rangle$
$\mathbf{1}$	$\langle 1, 0, 0, 0 \rangle$

Table 2: **Second-order expectation semiring** (variance semiring): Each element in the semiring is a **4-tuple** $\langle p, r, s, t \rangle$. The second and third rows define the operations between two elements $\langle p_1, r_1, s_1, t_1 \rangle$ and $\langle p_2, r_2, s_2, t_2 \rangle$, while the last two rows define the identities. Note that the multiplicative identity $\mathbf{1}$ has r,s and t components of 0.

P, R, S, T. "1" refers to the multiplicative identity of P. We continue to use distinguished symbols $\oplus, \otimes, \mathbf{0}, \mathbf{1}$ for the operations and identities in our "main semiring of interest," $\mathbb{E}_{P,R}$ or $\mathbb{E}_{P,R,S,T}$.

To compute equations (1)–(4) in this more general setting, we must still require multiplicative or additive decomposability, defining $p(d) \stackrel{\text{def}}{=} \prod_{e \in d} p_e$, $r(d) \stackrel{\text{def}}{=} \sum_{e \in d} r_e$, $s(d) \stackrel{\text{def}}{=} \sum_{e \in d} s_e$ as before. But the $\prod$ and $\sum$ operators here now denote appropriate operations within P, R, and S respectively (rather than the usual operations within $\mathbb{R}$).

4.2 Inside-Outside Speedup for First-Order Expectation Semirings

Under the first-order expectation semiring $\mathbb{E}_{\mathbb{R},\mathbb{R}^n}$, the inside algorithm of Figure 2 will return $\langle Z, \overline{r} \rangle$ where $\overline{r}$ is a vector of n feature expectations.

However, Eisner (2002, section 5) observes that this is inefficient when n is large. Why? The inside algorithm takes the trouble to compute an inside weight $\beta(v) \in \mathbb{R} \times \mathbb{R}^n$ for *each* node v in the hypergraph (or lattice). The second component of $\beta(v)$ is a presumably *dense* vector of all features that fire in all subderivations rooted at node v. Moreover, as $\beta(v)$ is computed in lines 3–8, that vector is built up (via the $\otimes$ and $\oplus$ operations of Table 1) as a linear combination of other dense vectors (the second components of the various $\beta(u)$). These vector operations can be slow.

A much more efficient approach (usually) is the traditional inside-outside algorithm (Baker, 1979).[7] Figure 4 generalizes the inside-outside algorithm to work with any expectation semiring $\mathbb{E}_{\mathbf{K},X}$.[8] We are given a hypergraph HG whose edges have weights $\langle k_e, x_e \rangle$ in this semiring (so now $k_e \in \mathbf{K}$ denotes only *part* of the edge weight, not all of it). INSIDE-OUTSIDE$(\text{HG}, \mathbf{K}, X)$ finds $\bigoplus_{d \in \text{D}} \bigotimes_{e \in d} \langle k_e, x_e \rangle$, which has the form $\langle \hat{k}, \hat{x} \rangle$.

But, INSIDE$(\text{HG}, \mathbb{E}_{\mathbf{K},X})$ could accomplish the same thing. So what makes the inside-outside algorithm more efficient? It turns out that $\hat{x}$ can be found quickly as a *single* linear combination $\sum_e \overline{k_e} x_e$ of just the feature vectors x_e that appear on *individual* hyperedges—typically a sum of very sparse vectors! And the linear coefficients $\overline{k_e}$, as well as $\hat{k}$, are computed entirely within the cheap semiring $\mathbf{K}$. They are based on β and α values obtained by first running INSIDE$(\text{HG}, \mathbf{K})$ and OUTSIDE$(\text{HG}, \mathbf{K})$, which use *only* the k_e part of the weights and ignore the more expensive x_e.

It is noteworthy that the expectation semiring is not used at all by Figure 4. Although the return value $\langle \hat{k}, \hat{x} \rangle$ is in the expectation semiring, it is built up not by $\oplus$ and $\otimes$ but rather by computing $\hat{k}$ and $\hat{x}$ separately. One might therefore wonder why the expectation semiring and its operations are still needed. One reason is that the input to Figure 4 consists of hyperedge weights $\langle k_e, x_e \rangle$ in the expectation semiring—and these weights may well have been constructed using $\otimes$ and $\oplus$. For example, Eisner (2002) uses finite-state operations such as composition, which do combine weights entirely within the expectation semiring before their result is passed to the forward-backward algorithm. A second reason is that when we work with a *second-order* expectation semiring in Section 4.4 below, the $\hat{k}, \beta$, and α values in Figure 4 will turn out to be elements of a first-order expectation semiring, and *they* must still be constructed by first-order $\otimes$ and $\oplus$, via calls to Figures 2–3.

Why does inside-outside work? Whereas the inside algorithm computes $\bigoplus_{d \in \text{D}} \bigotimes_{e \in d}$ in any semiring, the inside-outside algorithm exploits the special structure of an expectation semiring. By that semiring's definitions of $\oplus$ and $\otimes$ (Table 1), $\bigoplus_{d \in \text{D}} \bigotimes_{e \in d} \langle k_e, x_e \rangle$ can be found as

[7] Note, however, that the expectation semiring requires *only* the forward/inside pass to compute expectations, and thus it is more efficient than the traditional inside-outside algorithm (which requires two passes) if we are interested in computing only a small number of quantities.

[8] This follows Eisner (2002), who similarly generalized the forward-backward algorithm.

$\langle \sum_{d \in \mathrm{D}} \prod_{e \in d} k_e, \sum_{d \in \mathrm{D}} \sum_{e \in d} (\prod_{e' \in d, e' \neq e} k_{e'}) x_e \rangle$. The first component (giving $\hat{k}$) is found by calling the inside algorithm on just the k_e part of the weights. The second component (giving $\hat{x}$) can be rearranged into $\sum_e \sum_{d:\, e \in d} (\prod_{e' \in d, e' \neq e} k_{e'}) x_e = \sum_e \overline{k_e} x_e$, where $\overline{k_e} \stackrel{\text{def}}{=} \sum_{d:\, e \in d} (\prod_{e' \in d, e' \neq e} k_{e'})$ is found from β, α.

The application described at the start of this subsection is the classical inside-outside algorithm. Here $\langle k_e, x_e \rangle \stackrel{\text{def}}{=} \langle p_e, p_e r_e \rangle$, and the algorithm returns $\langle \hat{k}, \hat{x} \rangle = \langle Z, \overline{r} \rangle$. In fact, that $\hat{x} = \overline{r}$ can be seen directly: $\overline{r} = \sum_d p(d) r(d) = \sum_d p(d) (\sum_{e \in d} r_e) = \sum_e \sum_{d:\, e \in d} p(d) r_e = \sum_e (\overline{k_e} k_e) r_e = \sum_e \overline{k_e} x_e = \hat{x}$. This uses the fact that $\overline{k_e} k_e = \sum_{d:\, e \in d} p(d)$.

4.3 Lifting Trick for Second-Order Semirings

We now observe that the second-order expectation semiring $\mathbb{E}_{P,R,S,T}$ can be obtained indirectly by nesting one first-order expectation semiring inside another! First "lift" P to obtain the first-order expectation semiring $\mathbf{K} \stackrel{\text{def}}{=} \mathbb{E}_{P,R}$. Then lift this a second time to obtain the "nested" first-order expectation semiring $\mathbb{E}_{\mathbf{K},X} = \mathbb{E}_{(\mathbb{E}_{P,R}),(S \times T)}$, where we equip $X \stackrel{\text{def}}{=} S \times T$ with the operations $\langle s_1, t_1 \rangle + \langle s_2, t_2 \rangle \stackrel{\text{def}}{=} \langle s_1 + s_2, t_1 + t_2 \rangle$ and $\langle p, r \rangle \langle s, t \rangle \stackrel{\text{def}}{=} \langle ps, pt + rs \rangle$. The resulting first-order expectation semiring has elements of the form $\langle \langle p, r \rangle, \langle s, t \rangle \rangle$. Table 4 shows that it is indeed isomorphic to $\mathbb{E}_{P,R,S,T}$, with corresponding elements $\langle p, r, s, t \rangle$.

This construction of the second-order semiring as a first-order semiring is a useful bit of abstract algebra, because it means that known properties of first-order semirings will also apply to second-order ones. First of all, we are immediately guaranteed that the second-order semiring satisfies the semiring axioms. Second, we can directly apply the inside-outside algorithm there, as we now see.

4.4 Inside-Outside Speedup for Second-Order Expectation Semirings

Given a hypergraph weighted by a *second-order* expectation semiring $\mathbb{E}_{P,R,S,T}$. By recasting this as the *first-order* expectation semiring $\mathbb{E}_{\mathbf{K},X}$ where $\mathbf{K} = \mathbb{E}_{P,R}$ and $X = (S \times T)$, we can again apply INSIDE-OUTSIDE$(\mathrm{HG}, \mathbf{K}, X)$ to find the total weight of all derivations.

For example, to speed up Section 3.2, we may define $\langle k_e, x_e \rangle = \langle \langle p_e, p_e r_e \rangle, \langle p_e s_e, p_e r_e s_e \rangle \rangle$ for each hyperedge e. Then the inside-outside algorithm of Figure 4 will compute $\langle \hat{k}, \hat{x} \rangle = \langle \langle Z, \overline{r} \rangle, \langle \overline{s}, \overline{t} \rangle \rangle$, more quickly than the inside algorithm of Figure 2 computed $\langle Z, \overline{r}, \overline{s}, \overline{t} \rangle$.

Figure 4 in this case will run the inside and outside algorithms *in the semiring* $\mathbb{E}_{P,R}$, so that $k_e, \hat{k}, \alpha, \beta$, and $\overline{k_e}$ will now be elements of $P \times R$ (not just elements of P as in the first-order case). Finally it finds $\hat{x} = \sum_e \overline{k_e} x_e$, where $x_e \in S \times T$.[9]

This is a particularly effective speedup over the inside algorithm when R consists of scalars (or small vectors) whereas S, T are sparse high-dimensional vectors. We will see exactly this case in our experiments, where our weights $\langle p, r, s, t \rangle$ denote (probability, risk, gradient of probability, gradient of risk), or (probability, entropy, gradient of probability, gradient of entropy).

5 Finding Gradients on Hypergraphs

In Sections 3.2 and 4.1, we saw how our semirings helped find the sum Z of all $p(d)$, and compute *expectations* $\overline{r}, \overline{s}, \overline{t}$ of $r(d)$, $s(d)$, and $r(d)s(d)$.

It turns out that these semirings can *also* compute first- and second-order *partial derivatives* of all the above results, with respect to a parameter vector $\theta \in \mathbb{R}^m$. That is, we ask how they are affected when θ changes slightly from its current value. The elementary values p_e, r_e, s_e are now assumed to implicitly be functions of θ.

Case 1: Recall that $Z \stackrel{\text{def}}{=} \sum_d p(d)$ is computed by INSIDE$(\mathrm{HG}, \mathbb{R})$ if each hyperedge e has weight p_e. "Lift" this weight to $\langle p_e, \nabla p_e \rangle$, where $\nabla p_e \in \mathbb{R}^m$ is a gradient vector. Now $\langle Z, \nabla Z \rangle$ will be returned by INSIDE$(\mathrm{HG}, \mathbb{E}_{\mathbb{R}, \mathbb{R}^m})$— or, more efficiently, by INSIDE-OUTSIDE$(\mathrm{HG}, \mathbb{R}, \mathbb{R}^m)$.

Case 2: To differentiate a second time, "lift" the above weights again to obtain $\langle \langle p_e, \nabla p_e \rangle, \nabla \langle p_e, \nabla p_e \rangle \rangle = \langle \langle p_e, \nabla p_e \rangle, \langle \nabla p_e, \nabla^2 p_e \rangle \rangle$, where $\nabla^2 p_e \in \mathbb{R}^{m \times m}$ is the Hessian matrix of second-order mixed partial derivatives. These weights are in a second-order expectation semiring.[10] Now

[9] Figure 4 was already proved generally correct in Section 4.2. To understand more specifically how $\langle \overline{s}, \overline{t} \rangle$ gets computed, observe in analogy to the end of Section 4.2 that

$$\begin{aligned}
\langle \overline{s}, \overline{t} \rangle &= \textstyle\sum_d \langle p(d) s(d), p(d) r(d) s(d) \rangle \\
&= \textstyle\sum_d \langle p(d), p(d) r(d) \rangle \langle s(d), 0 \rangle \\
&= \textstyle\sum_d \langle p(d), p(d) r(d) \rangle \sum_{e \in d} \langle s_e, 0 \rangle \\
&= \textstyle\sum_e \sum_{d:\, e \in d} \langle p(d), p(d) r(d) \rangle \langle s_e, 0 \rangle \\
&= \textstyle\sum_e (\overline{k_e} k_e) \langle s_e, 0 \rangle = \sum_e \overline{k_e} \langle p_e, p_e r_e \rangle \langle s_e, 0 \rangle \\
&= \textstyle\sum_e \overline{k_e} \langle p_e s_e, p_e r_e s_e \rangle = \sum_e \overline{k_e} x_e = \hat{x}.
\end{aligned}$$

[10] Modulo the trivial isomorphism from $\langle \langle p, r \rangle, \langle s, t \rangle \rangle$ to $\langle p, r, s, t \rangle$ (see Section 4.3), the intended semiring both here and in Case 3 is the one that was defined at the start of Section 4.1, in which r, s are vectors and their product is defined

$$\begin{aligned}
\langle\langle p_1, r_1\rangle, \langle s_1, t_1\rangle\rangle \oplus \langle\langle p_2, r_2\rangle, \langle s_2, t_2\rangle\rangle &= \langle\langle p_1, r_1\rangle + \langle p_2, r_2\rangle,\ \langle s_1, t_1\rangle + \langle s_2, t_2\rangle\rangle \\
&= \langle\langle p_1 + p_2,\ r_1 + r_2\rangle,\ \langle s_1 + s_2,\ t_1 + t_2\rangle\rangle \\
\langle\langle p_1, r_1\rangle, \langle s_1, t_1\rangle\rangle \otimes \langle\langle p_2, r_2\rangle, \langle s_2, t_2\rangle\rangle &= \langle\langle p_1, r_1\rangle\langle p_2, r_2\rangle,\ \langle p_1, r_1\rangle\langle s_2, t_2\rangle + \langle p_2, r_2\rangle\langle s_1, t_1\rangle\rangle \\
&= \langle\langle p_1 p_2,\ p_1 r_2 + p_2 r_1\rangle,\ \langle p_1 s_2 + p_2 s_1,\ p_1 t_2 + p_2 t_1 + r_1 s_2 + r_2 s_1\rangle\rangle
\end{aligned}$$

Table 4: **Constructing second-order expectation semiring as first-order**. Here we show that the operations in $\mathbb{E}_{\mathbf{K},X}$ are isomorphic to Table 2's operations in $\mathbb{E}_{P,R,S,T}$, provided that $\mathbf{K} \stackrel{\text{def}}{=} \mathbb{E}_{P,R}$ and $X \stackrel{\text{def}}{=} S \times T$ is a $\mathbf{K}$-module, in which addition is defined by $\langle s_1, t_1\rangle + \langle s_2, t_2\rangle \stackrel{\text{def}}{=} \langle s_1 + s_2, t_1 + t_2\rangle$, and left-multiplication by $\mathbf{K}$ is defined by $\langle p, r\rangle\langle s, t\rangle \stackrel{\text{def}}{=} \langle ps, pt + rs\rangle$.

$\langle Z, \nabla Z, \nabla Z, \nabla^2 Z\rangle$ will be returned by INSIDE$(\text{HG}, \mathbb{E}_{\mathbb{R},\mathbb{R}^m,\mathbb{R}^m,\mathbb{R}^{m\times m}})$, or more efficiently by INSIDE-OUTSIDE$(\text{HG}, \mathbb{E}_{\mathbb{R},\mathbb{R}^m}, \mathbb{R}^m \times \mathbb{R}^{m\times m})$.

Case 3: Our experiments will need to find expectations and their partial derivatives. Recall that $\langle Z, \overline{r}\rangle$ is computed by INSIDE$(\text{HG}, \mathbb{E}_{\mathbb{R},\mathbb{R}^n})$ when the edge weights are $\langle p_e, p_e r_e\rangle$ with $r_e \in \mathbb{R}^n$. Lift these weights to $\langle\langle p_e, p_e r_e\rangle, \nabla\langle p_e, p_e r_e\rangle\rangle = \langle\langle p_e, p_e r_e\rangle, \langle \nabla p_e, (\nabla p_e) r_e + p_e(\nabla r_e)\rangle\rangle$. Now $\langle Z, \overline{r}, \nabla Z, \nabla\overline{r}\rangle$ will be returned by INSIDE$(\text{HG}, \mathbb{E}_{\mathbb{R},\mathbb{R}^n,\mathbb{R}^m,\mathbb{R}^{n\times m}})$ or by INSIDE-OUTSIDE$(\text{HG}, \mathbb{E}_{\mathbb{R},\mathbb{R}^n}, \mathbb{R}^m \times \mathbb{R}^{n\times m})$.[11]

5.1 What Connects Gradients to Expectations?

In ***Case 1***, we claimed that the *same* algorithm will compute either gradients $\langle Z, \nabla Z\rangle$ or expectations $\langle Z, \overline{r}\rangle$, if the hyperedge weights are set to $\langle p_e, \nabla p_e\rangle$ or $\langle p_e, p_e r_e\rangle$ respectively.[12] This may seem wonderful and mysterious. We now show in two distinct ways why this follows from our setup of Section 3.1. At the end, we derive as a special case the well-known relationship between gradients and expectations in log-linear models.

From Expectations to Gradients One perspective is that our semiring fundamentally finds expectations. Thus, we must be finding ∇Z by formulating it as a certain expectation $\overline{r}$. Specifically, $\nabla Z = \nabla \sum_d p(d) = \sum_d \nabla p(d) = \sum_d p(d) r(d) = \overline{r}$, provided that $r(d) = (\nabla p(d))/p(d)$. That can be arranged by defining $r_e \stackrel{\text{def}}{=} (\nabla p_e)/p_e$.[13] So that is why the input weights $\langle p_e, p_e r_e\rangle$ take the form $\langle p_e, \nabla p_e\rangle$.

From Gradients to Expectations An alternative perspective is that our semiring fundamentally finds gradients. Indeed, pairs like $\langle p, \nabla p\rangle$ have long been used for this purpose (Clifford, 1873) under the name "dual numbers." Operations on dual numbers, including those in Table 1, compute a result in $\mathbb{R}$ *along with* its gradient. For example, our $\otimes$ multiplies dual numbers, since $\langle p_1, \nabla p_1\rangle \otimes \langle p_2, \nabla p_2\rangle = \langle p_1 p_2, p_1(\nabla p_2) + (\nabla p_1) p_2\rangle = \langle p_1 p_2, \nabla(p_1 p_2)\rangle$. The inside algorithm thus computes both Z and ∇Z in a single "forward" or "inside" pass—known as automatic differentiation in the forward mode. The inside-outside algorithm instead uses the reverse mode (a.k.a. back-propagation), where a separate "backward" or "outside" pass is used to compute ∇Z.

How can we modify this machinery to produce expectations $\overline{r}$ given some *arbitrary* r_e of interest? Automatic differentiation may be used on any function (e.g., a neural net), but for our simple sum-of-products function Z, it happens that $\nabla Z = \nabla(\sum_d \prod_e p_e) = \sum_d \sum_{e\in d}(\prod_{e'\in d, e'\neq e} p_{e'})\nabla p_e$. Our trick is to surreptitiously *replace* the ∇p_e in the input weights $\langle p_e, \nabla p_e\rangle$ with $p_e r_e$. Then the output changes similarly: the algorithms will *instead* find $\sum_d \sum_{e\in d}(\prod_{e'\in d, e'\neq e} p_{e'}) p_e r_e$, which reduces to $\sum_d \sum_{e\in d} p(d) r_e = \sum_d p(d) \sum_{e\in d} r_e = \sum_d p(d) r(d) = \overline{r}$.

Log-linear Models as a Special Case Replacing ∇p_e with $p_e r_e$ is unnecessary if ∇p_e already equals $p_e r_e$. That is the case in log-linear models, where $p_e \stackrel{\text{def}}{=} \exp(r_e \cdot \theta)$ for some feature vector r_e associated with e. So there, ∇Z already equals $\overline{r}$—yielding a key useful property of log-linear

to be $rs^{\mathbf{T}}$, a matrix. However, when using this semiring to compute second derivatives (Case 2) or covariances, one may exploit the invariant that $r = s$, e.g., to avoid storing s and to compute $r_1 s_2 + s_1 r_2$ in multiplication simply as $2 \cdot r_1 r_2$.

[11] Or, if $n > m$, it is faster to instead use INSIDE-OUTSIDE$(\text{HG}, \mathbb{E}_{\mathbb{R},\mathbb{R}^m}, \mathbb{R}^n \times \mathbb{R}^{m\times n})$, swapping the second and third components of the 4-tuple and transposing the matrix in the fourth component. Algebraically, this changes nothing because $\mathbb{E}_{\mathbb{R},\mathbb{R}^n,\mathbb{R}^{m\times n\times m}}$ and $\mathbb{E}_{\mathbb{R},\mathbb{R}^m,\mathbb{R}^{n\times m\times n}}$ are isomorphic, thanks to symmetries in Table 2. This method computes the expectation of the gradient rather than the gradient of the expectation—they are equal.

[12] ***Cases 2–3*** relied on the fact that this relationship still holds even when the scalars $Z, p_e \in \mathbb{R}$ are replaced by more complex objects that we wish to differentiate. Our discussion below sticks to the scalar case for simplicity, but would generalize fairly straightforwardly. Pearlmutter and Siskind (2007) give the relevant generalizations of dual numbers.

[13] Proof: $r(d) = \sum_{e\in d} r_e = \sum_{e\in d}(\nabla p_e)/p_e = \sum_{e\in d} \nabla \log p_e = \nabla \sum_{e\in d} \log p_e = \nabla \log \prod_{e\in d} p_e = \nabla \log p(d) = (\nabla p(d))/p(d)$.

models, that $\nabla \log Z = (\nabla Z)/Z = \bar{r}/Z$, the vector of feature expectations (Lau et al., 1993).

6 Practical Applications

Given a hypergraph HG whose hyperedges e are annotated with values p_e. Recall from Section 3.1 that this defines a probability distribution over all derivations d in the hypergraph, namely $p(d)/Z$ where $p(d) \stackrel{\text{def}}{=} \prod_{e \in d} p_e$.

6.1 First-Order Expectation Semiring $\mathbb{E}_{\mathbb{R},\mathbb{R}}$

In Section 3, we show how to compute the **expected hypothesis length** or **expected feature counts**, using the algorithm of Figure 2 with a first-order expectation semiring $\mathbb{E}_{\mathbb{R},\mathbb{R}}$. In general, given hyperedge weights $\langle p_e, p_e r_e \rangle$, the algorithm computes $\langle Z, \bar{r} \rangle$ and thus $\bar{r}/Z$, the expectation of $r(d) \stackrel{\text{def}}{=} \sum_{e \in d} r_e$. We now show how to compute a few other quantities by choosing r_e appropriately.

Entropy on a Hypergraph The entropy of the distribution of *derivations* in a hypergraph[14] is

$$\begin{aligned} \mathrm{H}(p) &= -\sum_{d \in \mathrm{D}} (p(d)/Z) \log(p(d)/Z) \qquad (5) \\ &= \log Z - \frac{1}{Z} \sum_{d \in \mathrm{D}} p(d) \log p(d) \\ &= \log Z - \frac{1}{Z} \sum_{d \in \mathrm{D}} p(d) r(d) = \log Z - \frac{\bar{r}}{Z} \end{aligned}$$

provided that we define $r_e \stackrel{\text{def}}{=} \log p_e$ (so that $r(d) = \sum_{e \in d} r_e = \log p(d)$). Of course, we can compute $\langle Z, \bar{r} \rangle$ as explained in Section 3.2.

Cross-Entropy and KL Divergence We may be interested in computing the cross-entropy or KL divergence between two distributions p and q. For example, in variational decoding for machine translation (Li et al., 2009b), p is a distribution represented by a hypergraph, while q, represented by a finite state automaton, is an approximation to p. The cross entropy between p and q is defined as

$$\begin{aligned} \mathrm{H}(p,q) &= -\sum_{d \in \mathrm{D}} (p(d)/Z_p) \log(q(d)/Z_q) \qquad (6) \\ &= \log Z_q - \frac{1}{Z_p} \sum_{d \in \mathrm{D}} p(d) \log q(d) \\ &= \log Z_q - \frac{1}{Z_p} \sum_{d \in \mathrm{D}} p(d) r(d) = \log Z_q - \frac{\bar{r}}{Z_p} \end{aligned}$$

where the first term Z_q can be computed using the inside algorithm with hyperedge weights q_e, and the numerator and denominator of the second term using an expectation semiring with hyperedge weights $\langle p_e, p_e r_e \rangle$ with $r_e \stackrel{\text{def}}{=} \log q_e$.

The KL divergence to p from q can be computed as $\mathrm{KL}(p \,\|\, q) = \mathrm{H}(p,q) - \mathrm{H}(p)$.

Expected Loss (Risk) Given a reference sentence y^*, the expected loss (i.e., Bayes risk) of the hypotheses in the hypergraph is defined as,

$$\mathrm{R}(p) = \sum_{d \in \mathrm{D}} (p(d)/Z) L(\mathrm{Y}(d), y^*) \qquad (7)$$

where $\mathrm{Y}(d)$ is the target yield of d and $L(y, y^*)$ is the **loss** of the hypothesis y with respect to the reference y^*. The popular machine translation metric, BLEU (Papineni et al., 2001), is not **additively** decomposable, and thus we are not able to compute the expected loss for it. Tromble et al. (2008) develop the following loss function, of which a linear approximation to BLEU is a special case,

$$L(y, y^*) = -(\theta_0 |y| + \sum_{w \in N} \theta_w \#_w(y) \delta_w(y*)) \qquad (8)$$

where w is an n-gram type, N is a set of n-gram types with $n \in [1,4]$, $\#_w(y)$ is the number of occurrence of the n-gram w in y, $\delta_w(y^*)$ is an indicator to check if y^* contains at least one occurrence of w, and θ_n is the weight indicating the relative importance of an n-gram match. If the hypergraph is *already* annotated with n-gram ($n \geq 4$) language model states, this loss function is **additively** decomposable. Using $r_e \stackrel{\text{def}}{=} L_e$ where L_e is the *loss* for a hyperedge e, we compute the expected loss,

$$\mathrm{R}(p) = \frac{\sum_{d \in \mathrm{D}} p(d) L(\mathrm{Y}(d), y^*)}{Z} = \frac{\bar{r}}{Z} \qquad (9)$$

6.2 Second-Order Expectation Semirings

With second-order expectation semirings, we can compute from a hypergraph the expectation and variance of hypothesis length; the feature expectation vector and covariance matrix; the Hessian (matrix of second derivatives) of Z; and the gradients of entropy and expected loss. The computations should be clear from earlier discussion. Below we compute gradient of entropy or Bayes risk.

Gradient of Entropy or Risk It is easy to see that the gradient of entropy (5) is

$$\nabla \mathrm{H}(p) = \frac{\nabla Z}{Z} - \frac{Z \nabla \bar{r} - \bar{r} \nabla Z}{Z^2} \qquad (10)$$

[14] Unfortunately, it is intractable to compute the entropy of the distribution over *strings* (each string's probability is a sum over several derivations). But Li et al. (2009b, section 5.4) do estimate the gap between derivational and string entropies.

We may compute $\langle Z, \overline{r}, \nabla Z, \nabla \overline{r} \rangle$ as explained in ***Case 3*** of Section 5 by using $k_e \stackrel{\text{def}}{=} \langle p_e, p_e r_e, \nabla p_e, (\nabla p_e) r_e + p_e \nabla r_e \rangle \stackrel{\text{def}}{=} \langle p_e, p_e \log p_e, \nabla p_e, (1 + \log p_e) \nabla p_e \rangle$, where ∇p_e depends on the particular parameterization of the model (see Section 7.1 for an example).

Similarly, the gradient of risk of (9) is

$$\nabla \mathrm{R}(p) = \frac{Z \nabla \overline{r} - \overline{r} \nabla Z}{Z^2} \tag{11}$$

We may compute $\langle Z, \overline{r}, \nabla Z, \nabla \overline{r} \rangle$ using $k_e \stackrel{\text{def}}{=} \langle p_e, p_e L_e, \nabla p_e, L_e \nabla p_e \rangle$.

7 Minimum-Risk Training for MT

We now show how we improve the training of a Hiero MT model by optimizing an objective function that includes entropy and risk. Our objective function could be computed with a first-order expectation semiring, but computing it *along with its gradient* requires a second-order one.

7.1 The Model p

We assume a **globally normalized linear model** for its simplicity. Each derivation d is scored by

$$\text{score}(d) \stackrel{\text{def}}{=} \Phi(d) \cdot \theta = \sum_i \Phi_i(d)\, \theta_i \tag{12}$$

where $\Phi(d) \in \mathbb{R}^m$ is a vector of features of d. We then define the unnormalized distribution $p(d)$ as

$$p(d) = \exp(\gamma \cdot \text{score}(d)) \tag{13}$$

where the scale factor γ adjusts how sharply the distribution favors the highest-scoring hypotheses.

7.2 Minimum-Risk Training

Adjusting θ or γ changes the distribution p. Minimum error rate training (MERT) (Och, 2003) tries to tune θ to minimize the BLEU loss of a decoder that chooses the most probable output according to p. (γ has no effect.) MERT's specialized line-search addresses the problem that this objective function is piecewise constant, but it does not scale to a large number of parameters.

Smith and Eisner (2006) instead propose a *differentiable* objective that can be optimized by gradient descent: the Bayes risk $\mathrm{R}(p)$ of (7). This is the *expected* loss if one were (hypothetically) to use a *randomized* decoder, which chooses a hypothesis d in proportion to its probability $p(d)$. If entropy $\mathrm{H}(p)$ is large (e.g., small γ), the Bayes risk is smooth and has few local minima. Thus, Smith and Eisner (2006) try to avoid local minima by starting with large $\mathrm{H}(p)$ and decreasing it gradually during optimization. This is called **deterministic annealing** (Rose, 1998). As $\mathrm{H}(p) \to 0$ (e.g., large γ), the Bayes risk does approach the MERT objective (i.e. minimizing 1-best error).The objective is

$$\text{minimize } \mathrm{R}(p) - T \cdot \mathrm{H}(p) \tag{14}$$

where the "temperature" T starts high and is explicitly decreased as optimization proceeds.

7.3 Gradient Descent Optimization

Solving (14) for a given T requires computing the entropy $\mathrm{H}(p)$ and risk $\mathrm{R}(p)$ and their gradients with respect to θ and γ. Smith and Eisner (2006) followed MERT in constraining their decoder to only an n-best list, so for them, computing these quantities did not involve dynamic programming. We compare those methods to *training on a hypergraph containing exponentially many hypotheses*. In this condition, we need our new second-order semiring methods and must also approximate BLEU (during training only) by an additively decomposable loss (Tromble et al., 2008).[15]

Our algorithms require that $p(d)$ of (13) is **multiplicatively** decomposable. It suffices to define $\Phi(d) \stackrel{\text{def}}{=} \sum_{e \in d} \Phi_e$, so that all features are *local* to individual hyperedges; the vector Φ_e indicates which features fire on hyperedge e. Then $\text{score}(d)$ of (12) is **additively** decomposable:

$$\text{score}(d) = \sum_{e \in d} \text{score}_e = \sum_{e \in d} \Phi_e \cdot \theta \tag{15}$$

We can then set $p_e = \exp(\gamma \cdot \text{score}_e)$, and $\nabla p_e = \gamma p_e \Phi(e)$, and use the algorithms described in Section 6 to compute $\mathrm{H}(p)$ and $\mathrm{R}(p)$ and their gradients with respect to θ and γ.[16]

[15] Pauls et al. (2009) concurrently developed a method to maximize the *expected* n-gram counts on a *hypergraph* using gradient descent. Their objective is similar to the minimum risk objective (though without annealing), and their gradient descent optimization involves in algorithms in computing expected feature/n-gram counts as well as expected *products* of features and n-gram counts, which can be viewed as instances of our general algorithms with first- and second-order semirings. They focused on tuning only a small number (i.e. *nine*) of features as in a regular MERT setting, while our experiments involve both a small and a large number of features.

[16] It is easy to verify that the gradient of a function f (e.g. entropy or risk) with respect to γ can be written as a weighted sum of gradients with respect to the feature weights θ_i, i.e.

$$\frac{\partial f}{\partial \gamma} = \frac{1}{\gamma} \sum_i \theta_i \times \frac{\partial f}{\partial \theta_i} \tag{16}$$

7.4 Experimental Results

7.4.1 Experimental Setup

We built a translation model on a corpus for IWSLT 2005 Chinese-to-English translation task (Eck and Hori, 2005), which consists of 40k pairs of sentences. We used a *5-gram* language model with modified Kneser-Ney smoothing, trained on the bitext's English using SRILM (Stolcke, 2002).

7.4.2 Tuning a Small Number of Features

We first investigate how minimum-risk training (MR), with and without deterministic annealing (DA), performs compared to regular MERT. MR without DA just fixes $T = 0$ and $\gamma = 1$ in (14). All MR or MR+DA uses an approximated BLEU (Tromble et al., 2008) (for training only), while MERT uses the exact corpus BLEU in training.

The first five rows in Table 5 present the results by tuning the weights of *five* features ($\theta \in \mathbb{R}^5$). We observe that MR or MR+DA performs worse than MERT *on the dev set*. This may be mainly because MR or MR+DA uses an approximated BLEU while MERT doesn't. *On the test set*, MR or MR+DA on an n-best list is comparable to MERT. But our new approach, MR or MR+DA *on a hypergraph*, does consistently better (statistically significant) than MERT, despite approximating BLEU.[17]

Did DA help? For both n-best and hypergraph, MR+DA did obtain a better BLEU score than plain MR on the dev set.[18] This shows that DA helps with the local minimum problem, as hoped. However, DA's improvement on the dev set did not transfer to the test set.

7.4.3 Tuning a Large Number of Features

MR (with or without DA) is scalable to tune a large number of features, while MERT is not. To achieve competitive performance, we adopt a *forest reranking* approach (Li and Khudanpur, 2009; Huang, 2008). Specifically, our training has two stages. In the *first* stage, we train a baseline system as usual. We also find the optimal feature weights for the five features mentioned before, using the method of MR+DA operating on a hypergraph. In the *second* stage, we generate a hypergraph for each sentence in the training data (which consists of about 40k sentence pairs), using the baseline system. In this stage, we add 21k additional unigram and bigram target-side language model features (cf. Li and Khudanpur (2008)). For example, a specific bigram "the cat" can be a feature. Note that the total score by the baseline system is also a feature in the second-stage model. With these features and the 40k hypergraphs, we run the MR training to obtain the optimal weights.

	Training scheme	dev	test
	MERT (Nbest, small)	42.6	47.7
	MR (Nbest, small)	40.8	47.7
	MR+DA (Nbest, small)	41.6	47.8
NEW!	MR (hypergraph, small)	41.3	48.4
NEW!	MR+DA (hypergraph, small)	41.9	48.3
NEW!	MR (hypergraph, large)	42.3	**48.7**

Table 5: BLEU scores on the Dev and test sets under different training scenarios. In the "small" model, five features (i.e., one for the language model, three for the translation model, and one for word penalty) are tuned. In the "large" model, 21k additional unigram and bigram features are used.

During test time, a similar procedure is followed. For a given test sentence, the baseline system first generates a hypergraph, and then the hypergraph is reranked by the second-stage model. The last row in Table 5 reports the BLEU scores. Clearly, adding more features improves (statistically significant) the case with only five features. We plan to incorporate more informative features described by Chiang et al. (2009).[19]

8 Conclusions

We presented first-order expectation semirings and inside-outside computation in more detail than (Eisner, 2002), and developed extensions to higher-order expectation semirings. This enables efficient computation of many interesting quantities over the exponentially many derivations encoded in a hypergraph: second derivatives (Hessians), expectations of products (covariances), and expectations such as risk and entropy along with their derivatives. To our knowledge, algorithms for these problems have not been presented before.

Our approach is theoretically elegant, like other work in this vein (Goodman, 1999; Lopez, 2009; Gimpel and Smith, 2009). We used it practically to enable a new form of minimum-risk training that improved Chinese-English MT by 1.0 BLEU point. Our implementation will be released within the open-source MT toolkit **Joshua** (Li et al., 2009a).

[17] Pauls et al. (2009) concurrently observed a similar pattern (i.e., MR performs worse than MERT on the dev set, but performs better on a test set).

[18] We also verified that MR+DA found a better objective value (i.e., expected loss on the dev set) than MR.

[19] Their MIRA training tries to favor a specific oracle translation—indeed a specific tree—from the (pruned) hypergraph. MR does not commit to such an arbitrary choice.

References

J. K. Baker. 1979. Trainable grammars for speech recognition. In Jared J. Wolf and Dennis H. Klatt, editors, *Speech Communication Papers Presented at the 97th Meeting of the Acoustical Society of America*, MIT, Cambridge, MA, June.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *NAACL*, pages 218–226.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

W. K. Clifford. 1873. Preliminary sketch of biquaternions. *Proceedings of the London Mathematical Society*, 4:381–395.

Matthias Eck and Chiori Hori. 2005. Overview of the iwslt 2005 evaluation campaign. In *In Proc. of the International Workshop on Spoken Language Translation*.

Jason Eisner, Eric Goldlust, and Noah A. Smith. 2005. Compiling comp ling: practical weighted dynamic programming and the dyna language. In *HLT/EMNLP*, pages 281–290.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *ACL*, pages 1–8.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *ACL*, pages 205–208.

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

Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. 1993. Directed hypergraphs and applications. *Discrete Appl. Math.*, 42(2-3):177–201.

Kevin Gimpel and Noah A. Smith. 2009. Cube summing, approximate inference with non-local features, and dynamic programming without semirings. In *EACL*, pages 318–326.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Y Grandvalet and Y Bengio. 2004. Semi-supervised learning by entropy minimization. In *NIPS*, pages 529–536.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *IWPT*, pages 53–64.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594.

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *ACL*, pages 209–216.

Dan Klein and Christopher D. Manning. 2004. Parsing and hypergraphs. *New developments in parsing technology*, pages 351–372.

Raymond Lau, Ronald Rosenfeld, and Salim Roukos. 1993. Adaptive language modelling using the maximum entropy principle. In *Proc. ARPA Human Language Technologies Workshop*, pages 81–86.

Zhifei Li and Sanjeev Khudanpur. 2008. Large-scale discriminative n-gram language models for statistical machine translation. In *AMTA*, pages 133–142.

Zhifei Li and Sanjeev Khudanpur. 2009. Forest reranking for machine translation with the perceptron algorithm. In *GALE book chapter on "MT From Text"*.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar. Zaidan. 2009a. Joshua: An open source toolkit for parsing-based machine translation. In *WMT09*, pages 26–30.

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009b. Variational decoding for statistical machine translation. In *ACL*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *ACL*, pages 609–616.

Adam Lopez. 2009. Translation as weighted deduction. In *EACL*, pages 532–540.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: A method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Adam Pauls, John DeNero, and Dan Klein. 2009. Consensus training for consensus decoding in machine translation. In *EMNLP*.

B. A. Pearlmutter and J. M. Siskind. 2007. Lazy multivariate higher-order forward-mode ad. In *Proceedings of the 34th Annual Symposium on Principles of Programming Languages (POPL)*, pages 155–160.

Fernando C. N. Pereira and David H. D. Warren. 1983. Parsing as deduction. In *ACL*, pages 137–144.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: syntactically informed phrasal smt. In *ACL*, pages 271–279.

Kenneth Rose. 1998. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. In *Proceedings of the IEEE*, pages 2210–2239.

Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. 1994. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *ACL*, pages 787–794.

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

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice minimum-Bayes-risk decoding for statistical machine translation. In *EMNLP*, pages 620–629.

Feasibility of Human-in-the-loop Minimum Error Rate Training

Omar F. Zaidan and **Chris Callison-Burch**
Dept. of Computer Science, Johns Hopkins University
Baltimore, MD 21218, USA
{ozaidan,ccb}@cs.jhu.edu

Abstract

Minimum error rate training (MERT) involves choosing parameter values for a machine translation (MT) system that maximize performance on a tuning set as measured by an automatic evaluation metric, such as BLEU. The method is best when the system will eventually be evaluated using the same metric, but in reality, most MT evaluations have a human-based component. Although performing MERT with a human-based metric seems like a daunting task, we describe a new metric, RYPT, which takes human judgments into account, but only requires human input to build a database that can be reused over and over again, hence eliminating the need for human input at tuning time. In this investigative study, we analyze the diversity (or lack thereof) of the candidates produced during MERT, we describe how this redundancy can be used to our advantage, and show that RYPT is a better predictor of translation quality than BLEU.

1 Introduction

Many state-of-the-art machine translation (MT) systems over the past few years (Och and Ney, 2002; Koehn et al., 2003; Chiang, 2007; Koehn et al., 2007; Li et al., 2009) rely on several models to evaluate the "goodness" of a given candidate translation in the target language. The MT system proceeds by searching for the highest-scoring candidate translation, as scored by the different model components, and returns that candidate as the hypothesis translation. Each of these models need not be a probabilistic model, and instead corresponds to a feature that is a function of a (candidate translation,foreign sentence) pair.

Treated as a log-linear model, we need to assign a weight for each of the features. Och (2003) shows that setting those weights should take into account the evaluation metric by which the MT system will eventually be judged. This is achieved by choosing the weights so as to maximize the performance of the MT system on a development set, as measured by that evaluation metric. The other insight of Och's work is that there exists an efficient algorithm to find such weights. This process has come to be known as the MERT phase (for **M**inimum **E**rror **R**ate **T**raining) in training pipelines of MT systems.

A problem arises if the performance of the system is not judged by an automatic evaluation metric such as BLEU or TER, but instead through an evaluation process involving a human. The GALE evaluation, for instance, judges the quality of systems as measured by human-targeted TER (HTER), which computes the edit distance between the system's output and a version of the output post-edited by a human. The IWSLT and WMT workshops also have a manual evaluation component, as does the NIST Evaluation, in the form of adequacy and fluency (LDC, 2005).

In theory, one could imagine trying to optimize a metric like HTER during the MERT phase, but that would require the availability of an HTER automatic scorer, which, by definition, does not exist. If done manually, the scoring of thousands of candidates produced during MERT would literally take weeks, and cost a large sum of money. For these reasons, researchers resort to optimizing an automatic metric (almost always BLEU) as a proxy for human judgment.

As daunting as such a task seems for any human-based metric, we describe a new metric, RYPT, that takes human judgment into accout when scoring candidates, but takes advantage of the redundancy in the candidates produced during MERT. In this investigative study, we describe how this redundancy can be used to our advantage to eliminate the need to involve a human at any

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 52–61,
Singapore, 6-7 August 2009.

time except when building a database of reusable judgments, and furthermore show that RYPT is a better predictor of translation quality than BLEU, making it an excellent candidate for MERT tuning.

The paper is organized as follows. We start by describing the core idea of MERT before introducing our new metric, RYPT, and describing the data collection effort we undertook to collect the needed human judgments. We analyze a MERT run optimizing BLEU to quantify the level of redundancy in the candidate set, and also provide an extensive analysis of the collected judgments, before describing a set of experiments showing RYPT is a better predictor of translation quality than BLEU. Following a discussion of our findings, we briefly review related work, before pointing out future directions and summarizing.

2 Och's Line Search Method

A common approach to translating a source sentence f in a foreign language is to select the candidate translation e that maximizes the posterior probability:

$$Pr(e \mid f) \stackrel{\text{def}}{=} \frac{\exp(s_\Lambda(e,f))}{\sum_{e'} \exp(s_\Lambda(e',f))}.$$

This defines $Pr(e \mid f)$ using a *log-linear model* that associates a sentence pair (e, f) with a feature vector $\Phi(e,f) = \{\phi_1(e,f), ..., \phi_M(e,f)\}$, and assigns a score

$$s_\Lambda(e,f) \stackrel{\text{def}}{=} \Lambda \cdot \Phi(e,f) = \sum_{m=1}^{M} \lambda_m \phi_m(e,f)$$

for that sentence pair, with the feature weights $\Lambda = \{\lambda_1, ..., \lambda_M\}$ being the parameters of the model. Therefore, the system selects the translation $\hat{e}$:

$$\hat{e} = \operatorname*{argmax}_e Pr(e \mid f) = \operatorname*{argmax}_e s_\Lambda(e,f). \quad (1)$$

Och (2003) provides evidence that Λ should be chosen by optimizing an objective function basd on the evaluation metric of interest, rather than likelihood. Since the error surface is not smooth, and a grid search is too expensive, Och suggests an alternative, efficient, line optimization approach.

Assume we are performing a line optimization along the d^{th} dimension. Consider a foreign sentence f, and let the candidate set for f be $\{e_1, ..., e_K\}$. Recall from (1) that the 1-best candidate at a given Λ is the one with maximum $\sum_{m=1}^{M} \lambda_m \phi_m(e_k, f)$. We can rewrite the sum as $\lambda_d \phi_d(e_k, f) + \sum_{m \neq d} \lambda_m \phi_m(e_k, f)$. The second term is constant with respect to λ_d, and so is $\phi_d(e_k, f)$. Renaming those two quantities $\texttt{offest}_\Lambda(e_k)$ and $\texttt{slope}(e_k)$, we get

$$s_\Lambda(e_k, f) = \texttt{slope}(e_k)\lambda_d + \texttt{offset}_\Lambda(e_k).$$

Therefore, if we plot the score for a candidate translation vs. λ_d, that candidate will be represented by a line. If we plot the lines for all candidates (Figure 1), then the upper envelope of these lines indicates the best candidate at any value for λ_d.

Therefore, the objective function is piece-wise linear across any of the M dimensions[1], meaning we only need to evaluate it at the "critical" points corresponding to line intersection points. Furthermore, we only need to calculate the sufficient statistics once, at the smallest critical point, and then simply adjust the sufficient statistics to reflect changes in the set of 1-best candidates.

2.1 The BLEU Metric

The metric most often used with MERT is BLEU (Papineni et al., 2002), where the score of a candidate c against a reference translation r is:

$$BLEU = BP(len(c), len(r)) \cdot \exp(\sum_{n=1}^{4} \frac{1}{4} \log p_n),$$

where p_n is the n-gram precision[2] and BP is a brevity penalty meant to penalize short outputs, to discourage improving precision at the expense of recall.

There are several compelling reasons to optimize to BLEU. It is the most widely reported metric in MT research, and has been shown to correlate well with human judgment (Papineni et al., 2002; Coughlin, 2003). But BLEU is also particularly suitable for MERT, because it can be computed quite efficiently, and its sufficient statistics are decomposable, as required by MERT.[3,4]

[1] Or, in fact, along any linear combination of the M dimensions.

[2] *Modifed* precision, to be precise, based on clipped n-gram counts.

[3] Note that for the sufficient statistics to be decomposable, the metric itself need not be – this is in fact the case with BLEU.

[4] Strictly speaking, the sufficient statistics need not be de-

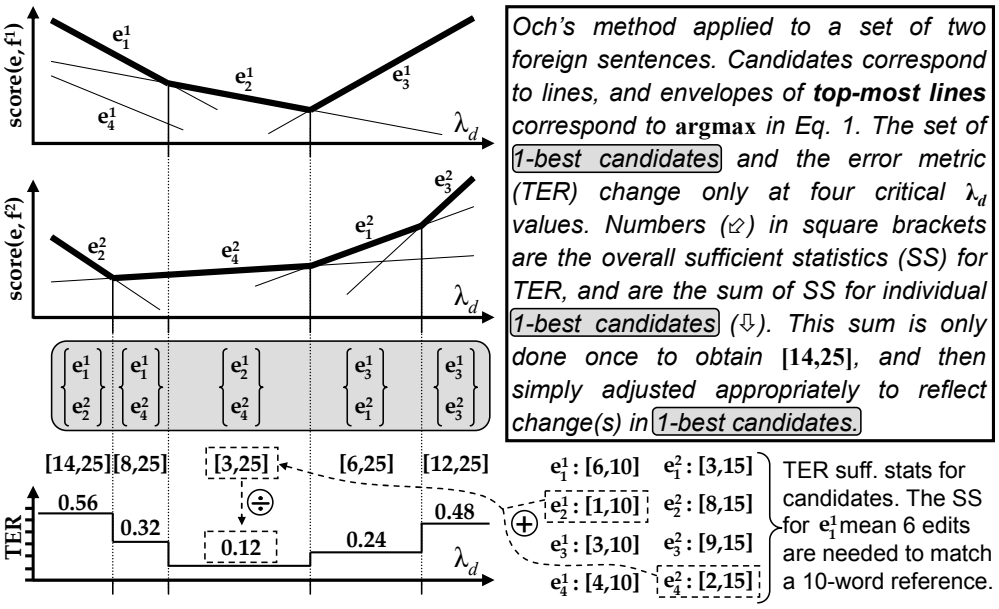


Figure 1: Och's method applied to a set of two foreign sentences. This figure is essentially a visualization of equation (1). We show here sufficient statistics for TER for simplicity, since there are only 2 of them, but the metric optimized in MERT is usually BLEU.

In spite of these advantages, recent work has pointed out a number of problematic aspects of BLEU that should cause one to pause and reconsider the reliance on it. Chiang et al. (2008) investigate several weaknesses in BLEU and show there are realistic scenraios where the BLEU score should not be trusted, and in fact behaves in a counter-intuitive manner. Furthermore, Callison-Burch et al. (2006) point out that it is not always appropriate to use BLEU to compare systems to each other. In particular, the quality of rule-based systems is usually underestimated by BLEU.

All this raises doubts regarding BLEU's adequacy as a proxy for human judgment, which is a particularly important issue in the context of setting parameters during the MERT phase. But what is the alternative?

2.2 (Non-)Applicability of Och's Method to Human Metrics

In principle, MERT is applicable to any evaluation metric, including HTER, as long as its sufficient statistics are decomposable.[4] In practice, of course, the method requires the evaluation of thousands of candidate translations. Whereas this is not a problem with a metric like BLEU, for which automatic (and fast) scorers are available, such an evaluation with a human metric would require a large amount of effort and money, meaning that a single MERT run would take weeks to complete, and would cost thousands of dollars. Assume a single candidate string takes 10 seconds to post-edit, at a cost of $0.10. Even with such an (overly) optimistic estimate, scoring 100 candidates for each of 1000 sentences would take 35 8-hour work days and cost $10,000. The cost would further grow linearly with the number of MERT iterations and the n-best list size. On the other hand, optimizing for BLEU takes on the order of minutes per iteration, and costs nothing.

[4] composable in MERT, as they can be recalculated at each critical point. However, this would slow down the optimization process quite a bit, since one cannot traverse the dimension by simply adjusting the sufficient statistics to reflect changes in 1-best candidates.

2.3 The RYPT Metric

We suggest here a new metric that combines the best of both worlds, in that it is based on human judgment, but that is a viable metric to be used in the MERT phase. The key to the feasiblity is the reliance on a *database* of human judgment rather than immendiate feedback for each candidate, and so human feedback is only needed once, and the collected human judgments can be reused over and over again by an automatic scorer.

The basic idea is to reward syntactic constituents in the source sentence that get aligned to "acceptable" translations in the candidate sen-

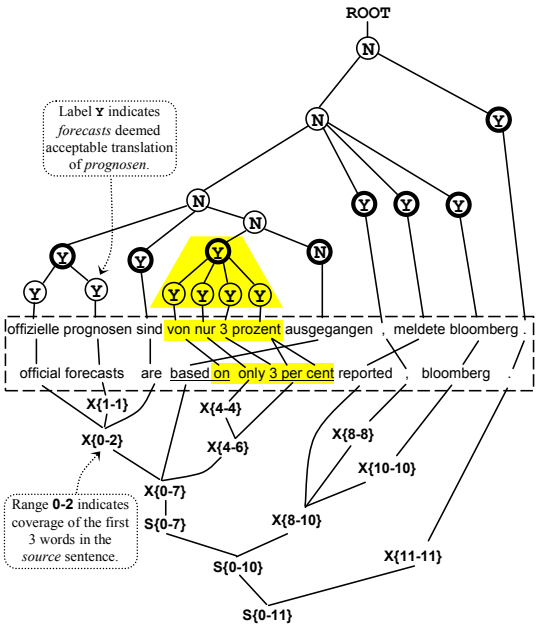


Figure 2: The source parse tree (top) and the candidate derivation tree (bottom). Nodes in the parse tree with a thick border correspond to the frontier node set with maxLen $= 4$. The human annotator only sees the portion surrounded by the dashed rectangle, including the highlighting (though excluding the word alignment links).

tence, and penalize constituents that do not. For instance, consider the source-candidate sentence pair of Figure 2. To evaluate the candidate translation, the source parse tree is first obtained (Dubey, 2005), and each subtree is matched with a substring in the candidate string. If the source substring covered by this subtree is translated into an acceptable substring in the candidate, that node gets a YES label. Otherwise, the node gets a NO label.

The metric we propose is taken to be the ratio of YES nodes in the parse tree (or RYPT). The candidate in Figure 2, for instance, would get a RYPT score of $13/18 = 0.72$.

To justify its use as a proxy for HTER-like metrics, we need to demonstrate that this metric correlates well with human judgment. But it is also important to show that we can obtain the YES/NO label assignments in an efficient and affordable manner. At first glance, this seems to require a human to provide judgments for each candidate, much like with HTER. But we describe in the next section strategies that minimize the number of judgments we need to actually collect.

3 Collecting Human Judgments

The first assumption we make to minimize the number of human judgments, is that once we have a judgment for a source-candidate substring pair, that same judgment can be used across all candidates for this source sentence. In other words, we build a database for each source sentence, which consists of <source substring,target substring,judgment> entries. For a given source substring, multiple entries exist, each with a different target candidate substring. The judgment field is one of YES, NO, and NOT SURE.

Note that the entries do not store the full candidate string, since we reuse a judgment across all the candidates of that source sentence. For instance, if we collect the judgment:

<der patient,the patient,YES>

from the sentence pair:

der patient wurde isoliert .
the patient was isolated .

then this would apply to any candidate translation of this source sentence. And so all of the following substrings are labeled YES as well:

the patient isolated .
the patient was in isolation .
the patient has been isolated .

Similarly, if we collect the judgment:

<der patient,of the patient,NO>

from the sentence pair:

der patient wurde isoliert .
of the patient was isolated .

then this would apply to any candidate translation of the source, and the following substrings are labeled NO as well:

of the patient isolated .
of the patient was in isolation .
of the patient has been isolated .

The strategy of using judgments across candidates reduces the amount of labels we need to collect, but evaluating a candidate translation for the source sentence of Figure 2 would still require obtaining 18 labels, one for each node in the parse tree. Instead of querying a human for each one

of those nodes, it is quite reasonable to percolate existing labels up and down the parse tree: if a node is labeled NO, this likely means that all its ancestors would also be labeled NO, and if a node is labeled YES, this likely means that all its descendents whould also be labeled YES.

While those two strategies (using judgments across candidates, and percolating labels up and down the tree) are only approximations for the true labels, employing them considerably reduces the amount of data we need to collect.

3.1 Obtaining Source-to-Candidate Alignments

How do we determine which segment of the candidate sentence aligns to a given source segment? Given a word alignment between the source and the candidate, we take the target substring to contain any word aligned with at least one word in the source segment. One could run an aligner (e.g. GIZA++) on the two sentences to obtain the word alignment, but we take a different approach.

We use Joshua (Li et al., 2009), in our experiments. Joshua is a hierarchical parsing-based MT system, and it can be instructed to produce derivation trees instead of the candidate sentence string itself. Furthermore, each node in the derivation tree is associated with the two indices in the source sentence that indicate the segment corresponding to this derivation subtree (the numbers indicated in curly brackets in Figure 2).

Using this information, we are able to recover most of the phrasal alignments. There are other phrasal alignments that can be deduced from the structure of the tree indirectly, by systematically discarding source words that are part of another phrasal alignment. For instance, in Figure 2, one can observe the alignment (offizielle,prognosen,sind)–(official,forecasts,are) and the alignment (prognosen)–(forecasts) to deduce (offizielle,sind)–(official,are).

Although some of the phrasal alignment are one-to-one mappings, many of them are many-to-many. By construction, any deduced many-to-many mapping has occurred in the training parallel corpus at least once. And so we recover the individual word alignments by consulting the parallel corpus from which the grammar rules were extracted (which requires maintaining the word alignments obtained prior to rule extraction).[5]

We emphasize here that our recovery of word alignment from phrasal alignment is independent from the hierarchical and parsing-based nature of the Joshua system. And so the alignment approach we suggest here can be applied to a different MT system as well, as long as that system provides phrasal alignment along with the output. In particular, a phrase-based system such as Moses can be modified in a straightforward manner to provide phrasal alignments, and then apply our method.

4 Data Collection

We chose the WMT08 German-English news dataset to work with, and since this is an investigative study of a novel approach, we collected judgments for a subset of 250 source sentences from the development set for the set of candidate sentences produced in the last iteration of a MERT run optimizing BLEU on the full 2051-sentence development set. The MT system we used is Joshua (Li et al., 2009), a software package that comes complete with a grammar extraction module and a MERT module, in addition to the decoder itself.

What segments of the source should be chosen to be judged? We already indicated that we limit ourselves, by definition of RYPT, to segments that are covered exactly by a subtree in the source parse tree. This has a couple of nice advantages: it allows us to present an annotator with a high number of alternatives judged simulataneously (since the annotator is shown a source segment and several candidates, not just one), and this probably also makes judging them easier – it is reasonable to assume that strings corresponding to syntactic constituents are easier to process by a human.

Our query selection strategy attempts to maximize the amount of YES/NO percolation that would take place. We therefore ensure that for any 2 queries, the corresponding source segments do not overlap: such overlap indicates that one subtree is completely contained within the other. Having both queries (in the same batch) might be redundant if we use the above percolation procedure.

The idea is to select source segments so that they fully cover the entire source sentence, but have no overlap amongst them. In one extreme, each query would correspond to an entire parse tree. This is not ideal since the overwhelming majority of the judgments will most likely be NO,

[5] We incorporated our implementation of the source-candidate aligner into the Joshua software as a new `aligner` package.

which does not help identify where the problem is. In the other extreme, each query would correspond to a subtree rooted at a preterminal. This is also not ideal, since it would place too much emphasis on translations of unigrams.

So we need a middle ground. We select a maximum-source-length `maxLen` to indicate how long we're willing to let source segments be. Then we start at the root of the parse tree, and propagate a "frontier" node set down the parse tree, to end up with a set of nodes that fully cover the source sentence, have no overlap amongst them, and with each covering no more than `maxLen` source words. For instance, with `maxLen` set to 4, the frontier set of Figure 2 are the nodes with a thick border. An algorithmic description is provided in Algorithm 1.

Algorithm 1 Constructing the frontier node set for a parse tree.

Input: A source parse tree T rooted at `ROOT`, and a maximum source length `maxLen`.
Return: A nonempty set `frontierSet`, containing a subset of the nodes in T.

1. Initialize `frontierSet` to the empty set.
2. Initialize `currNodes` to $\{$`ROOT`$\}$.
3. **while** `currNodes` is not empty **do**
4. Initialize `newNodes` to the empty set.
5. **for** each node N in `currNodes` **do**
6. **if** N covers $\leq$ `maxLen` source words **then**
7. Add N to `frontierSet`.
8. **else**
9. Add children of N to `newNodes`.
10. **end if**
11. **end for**
12. Set `currNodes` = `newNodes`
13. **end while**
14. Return `frontierSet`.

This would ensure that our queries cover between 1 and `maxLen` source words, and ensures they do not overlap, which would allow us to take full advantage of the downward-`YES` and upward-`NO` percolation. We set `maxLen` $= 4$ based on a pilot study of 10 source sentences and their candidates, having observed that longer segments tend to always be labeled as `NO`, and shorter segments tend to be so deep down the parse tree.

4.1 Amazon Mechanical Turk

We use the infrastructure of Amazon's Mechanical Turk (AMT)[6] to collect the labels. AMT is a virtual marketplace that allows "requesters" to create and post tasks to be completed by "workers" around the world. To create the tasks (called *Human Intelligence Tasks*, or HITs), a requester supplies an HTML template along with a comma-separated-values database, and AMT automatically creates the HITs and makes them available to workers. The queries are displayed as an HTML page (based on the provided HTML template), with the user indicating the label (`YES`, `NO`, or `NOT SURE`) by selecting the appropriate radio button. The instructions read, in part:[7]

> You are shown a "source" German sentence with a highlighted segment, followed by several candidate translations with corresponding highlighted segments. Your task is to decide if each highlighted English segment is an acceptable translation of the highlighted German segment.

In each HIT, the worker is shown up to 10 alternative translations of a highlighted source segment, with each itself highlighted within a full candidate string in which it appears. To aid the worker in the task, they are also shown the reference translation, with a highlighted portion that corresponds to the source segment, deduced using word alignments obtained with GIZA++.[8]

4.2 Cost of Data Collection

The total number of HITs created was 3873, with the reward for completing a HIT depending on how many alternative translations are being judged. On average, each HIT cost 2.1 cents and involved judging 3.39 alternatives. 115 distinct workers put in a total of 30.82 hours over a period of about 4 days. On average, a label required 8.4 seconds to determine (i.e. at a rate of 426 labels per hour). The total cost was $81.44: $21.43 for Amazon's commission, $53.47 for wages, and

[6] AMT's website: `http://www.mturk.com`.

[7] Template and full instructions can be viewed at `http://cs.jhu.edu/~ozaidan/hmert`.

[8] These alignments are not always precise, and we do note that fact in the instructions. We also deliberately highlight the reference substring in a different color to make it clear that workers should judge a candidate substring primarily based on the source substring, not the reference substring.

$6.54 for bonuses[9], for a cost per label of 0.62 cents (i.e. at a rate of 161.32 labels per dollar). Excluding Amazon's commission, the effective hourly 'wage' was $1.95.

5 Experimental Results and Analysis

By limiting our queries to source segments corresponding to frontier nodes with `maxLen` $= 4$, we obtain a total of 3601 subtrees across the 250 sentences, for an average of 14.4 per sentence. On average, each subtree has 3.65 alternative translations. Only about 4.8% of the judgments were returned as `NOT SURE` (or, occasionally, blank), with the rest split into 35.1% `YES` judgments and 60.1% `NO` judgments.

The coverage we get before percolating labels up and down the trees is 39.4% of the nodes, increasing to a coverage of 72.9% after percolation. This is quite good, considering we only do a single data collection pass, and considering that about 10% of the subtrees do not align to candidate substrings to begin with (e.g. single source words that lack a word alignment into the candidate string).

The main question, of course, is whether or not those labels allow us to calculate a RYPT score that is reliably correlated with human judgment. We designed an experiment to compare the predictive power of RYPT vs. BLEU. Given the candidate set of a source sentence, we rerank the candidate set according to RYPT and extract the top-1 candidate, and we rerank the candidate set according to BLEU, and extract the top-1 candidate. We then present the two candidates to human judges, and ask them to choose the one that is a more adequate translation. For reliability, we collect 3 judgments per sentence pair comparison, instead of just 1.

The results show that RYPT significantly outperforms BLEU when it comes to predicting human preference, with its choice prevailing in 46.1% of judgments vs. 36.0% for BLEU, with 17.9% judged to be of equal quality (left half of Table 1). This advantage is especially true when the judgments are grouped by sentence, and we examine cases of strong agreement among the three annotators (Table 2): whereas BLEU's candidate is strongly preferred in 32 of the candidate pairs (bottom 2 rows), RYPT's candidate is strongly preferred in about double that number: 60 candidate pairs (top 2 rows).

This is quite a remarkable result, given that BLEU, by definition, selects a candidate that has significant overlap with the reference shown to the annotators to aid in their decision-making. This means that BLEU has an inherent advantage in comparisons where both candidates are more or less of equal quality, since annotators are encouraged (in the instructions) to make a choice even if the two candidates seem of be of equal quality at first glance. Pressed to make such a choice, the annotator is likely to select the candidate that superficially 'looks' more like the reference to be the 'better' of the two candidates. That candidate will most likely be the BLEU-selected one.

To test this hypothesis, we repeated the experiment *without showing the annotators the reference translations*, and limited data collection to workers living in Germany, making judgments based only on the source sentences. (We only collected one judgment per source sentence, since German workers on AMT are in short supply.)

As expected, the difference is even more pronounced: human judges prefer the RYPT-selected candidate 45.2% of the time, while BLEU's candidate is preferred only 29.2% of the time, with 25.6% judged to be of equal quality (right half of Table 1). Our hypothesis is further supported by the fact that most of the gain of the "equal-quality" category comes from BLEU, which loses 6.8 percentage points, whereas RYPT's share remains largely intact, losing less than a single percentage point.

5.1 Analysis of Data Collection

Recall that we minimize data collection by performing label percolation and by employing a frontier node set selection strategy. While the results just presented indicate those strategies provide a good approximation of some 'true' RYPT score, label percolation was a strategy based primarily on intuition, and choosing `maxLen` $= 4$ for frontier set construction was based on examining a limited amount of preliminary data.

Therefore, and in addition to encouraging empricial results, we felt a more rigorous quantitative analysis was in order, especially with future, more ambitious annotation projects on the horizon. To this end, we collected a *complete* set of judgments for 50 source sentences and their candidates. That is, we generated a query for each and every node

[9] We would review the collected labels and give a 20% reward for good workers to encourage them to come back and complete more HITs.

	References shown; unrestricted		References **not** shown; restricted to DE workers	
Preferred candidate	# judgments	% judgments	# judgments	% judgments
Top-1 by RYPT	346	46.1	113	45.2
Top-1 by BLEU	270	36.0	73	29.2
Neither	134	17.9	64	25.6
Total	750	100.0	250	100.0

Table 1: Ranking comparison results. The left half corresponds to the experiment (open to all workers) where the English reference was shown, whereas the right half corresponds to the experiment (open only to workers living in Germany) where the English reference was *not* shown.

Aggregate	# sentences	% sentences	Aggregate	# sentences	% sentences
RYPT +3	45	18.0			
RYPT +2	15	6.0	RYPT +any	120	48.0
RYPT +1	60	24.0			
$\pm$ 0	42	16.8	$\pm$ 0	42	16.8
BLEU +1	55	22.0			
BLEU +2	5	2.0	BLEU +any	88	35.2
BLEU +3	28	11.2			
Total	250	100.0	Total	250	100.0

Table 2: Ranking comparison results, grouped by sentence. This table corresponds to the left half of Table 1. 3 judgments were collected for each comparison, with the "aggregate" for a comparison calculated from these 3 judgments. For instance, an aggregate of "RYPT +3" means all 3 judgments favored RYPT's choice, and "RYPT +1" means one more judgment favored RYPT than did BLEU.

in the source parse tree, instead of limiting ourselves to a frontier node set. (Though we did limit the length of a source segment to be ≤ 7 words.) This would allow us to judge the validity of label percolation, and under different `maxLen` values.

Furthermore, we collected *multiple* judgments for each query in order to minimize the effet of bad/random annotations. For each of 5580 generated queries, we collected five judgments, for a total of 27,900 judgments.[10] As before, the annotator would pick one of `YES`, `NO`, and `NOT SURE`.

First, collecting multiple judgments allowed us to investigate inter-annotator agreement. In 68.9% of the queries, at least 4 of the 5 annotators chose the same label, signifying a high degree of inter-annotator agreement. This is especially encouraging considering that we identified about 15% of the HITs as being of poor quality, and blocked the respective annotators from doing further HITs.[11]

We then examine the *applicability* and *validity* of label percolation. For each of 7 different values for Algorithm 1's `maxLen`, we ignore all but labels that would be requested under that `maxLen` value, and percolate the labels up and down the tree. In Figure 3 we plot the coverage before and after percolation (middle two curves), and observe expansion in coverage across different values of `maxLen`, peaking at about +33% for `maxLen`$= 4$ and 5, with most of the benefit coming from `YES` percolation (bottom two curves).

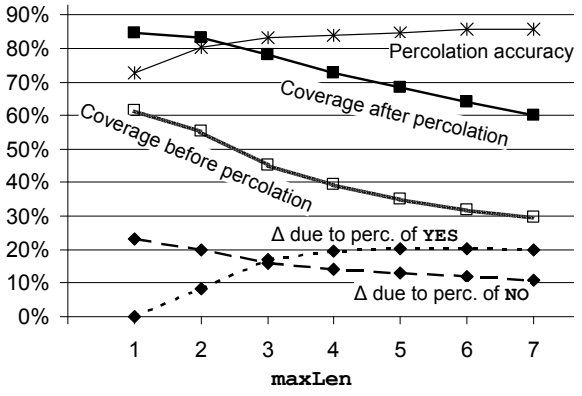


Figure 3: Label percolation under different `maxLen` values. The bottom two curves are the breakdown of the difference between the middle two. Accuracy is measured against majority votes.

[10] For a given query, the five collected judgments are from five different annotators, since AMT ensures an annotator is never shown the same HIT twice.

[11] It is especially easy to identify (and then block) such annotators when they submit a relatively large number of HITs, since inspecting some of their annotations would indicate they are answering randomly and/or inconsistently.

We also measure the accuracy of labels deduced from percolation (top curve of Figure 3). We define a percolated label to be correct if it matches the label given by a majority vote over the collected labels for that particular node. We find that accuracy at low `maxLen` values is significantly lower than at higer values (e.g. 72.6% vs. 84.1% for 1 vs. 4). This means a middle value such as 3 or 4 is optimal. Higher values could be suitable if we wish to emphasize translation fluency.

6 Related Work

Nießen et al. (2000) is an early work that also constructs a database of translations and judgments. There, a source sentence is stored along with all the translations that have already been manually judged, along with their scores. They utilize this database to carry out "semi-automatic" evaluation in a fast and convenient fashion thanks to tool they developed with a user-friendly GUI.

In their annual evaluation, the WMT workshop has effectively conducted manual evaluation of submitted systems over the past few years by distributing the work across tens of volunteers, though they relied on a self-designed online portal. On the other hand, Snow et al. (2008) illustrate how AMT can be used to collect data in a "fast and cheap" fashion, for a number of NLP tasks, such as word sense disambiguation. They go a step further and model the behavior of their annotators to reduce annotator bias. This was possible as they collect multiple judgments for each query from multiple annotators.

The question of how to design an automatic metric that best approximates human judgment has received a lot of attention lately. NIST started organizing the Metrics for Machine Translation Challenge (MetricsMATR) in 2008, with the aim of developing automatic evaluation metrics that correlate highly with human judgment of translation quality. The latest WMT workshop (Callison-Burch et al., 2009) also conducted a full assessment of how well a suite of automatic metrics correlate with human judgment.

7 Future Work

This pilot study has demonstrated the feasibility of collecting a large number of human judgments, and has shown that the RYPT metric is better than BLEU at picking out the best translation. The next step is to run a complete MERT run. This will involve collecting data for thousands of alternative translations for several hundreds source sentences. Based on our analysis, this it should be cost-effective to solicit these judgments using AMT. After training MERT using RYPT as an objective function the, the next logical step would be to compare two outputs of a system. One output would have parameters optimized to BLEU and the other to RYPT. The hope is that the RYPT-trained system would be better under the final HTER evaluation than the BLEU-trained system.

We are also investigating a probabilistic approach to percolating the labels up and down the tree, whereby the label of a node is treated as a random variable, and inference is performed based on values of the other observed nodes, as well as properties of the source/candidate segment. Cast this way, a probabilistic approach is actually quite appealing, and one could use collected data to train a prediction model (such as a Markov random field).

8 Summary

We propose a human-based metric, RYPT, that is quite feasible to optimize using MERT, relying on the redundancy in the candidate set, and collecting judgments using Amazon's Mechanical Turk infrastructure. We show this could be done in a quite cost-effective manner, and produces data of good quality. We show the effectiveness of the metric by illustrating that it is a better predictor of human judgment of translation quality than BLEU, the most commonly used metric in MT. We show this is the case even with a modest amount of data that does not cover the entirety of all parse trees, on which the metric is dependent. The collected data represents a database that can be reused over and over again, hence limiting human feedback to the initial phase only.

Acknowledgments

This research was supported by the EuroMatrix-Plus project funded by the European Commission (7th Framework Programme), by the Defense Advanced Research Projects Agency's GALE program under Contract No. HR0011-06-2-0001, and the US National Science Foundation under grant IIS-0713448. The views and findings are the authors' alone.

References

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of Bleu in machine translation research. In *Proceedings of EACL*, pages 249–256.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece, March. Association for Computational Linguistics.

David Chiang, Steve DeNeefe, Yee Seng Chan, and Hwee Tou Ng. 2008. Decomposability of translation metrics for improved evaluation and efficient algorithms. In *Proceedings of EMNLP*, pages 610–619.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Deborah Coughlin. 2003. Correlating automated and human assessments of machine translation quality. In *Proceedings of MT Summit IX*.

Amit Dubey. 2005. What to do when lexicalization fails: parsing German with suffix analysis and smoothing. In *Proceedings of ACL*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 127–133.

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

LDC. 2005. Linguistic data annotation specification: Assessment of fluency and adequacy in translations. Revision 1.5.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139.

Sonja Nießen, Franz Josef Och, Gregor Leusch, and Hermann Ney. 2000. An evaluation tool for machine translation: Fast evaluation for mt research. In *Proceedings of LREC*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*, pages 295–302.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

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

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*, pages 254–263.

Cube Pruning as Heuristic Search

Mark Hopkins and Greg Langmead
Language Weaver, Inc.
4640 Admiralty Way, Suite 1210
Marina del Rey, CA 90292
{mhopkins,glangmead}@languageweaver.com

Abstract

Cube pruning is a fast inexact method for generating the items of a beam decoder. In this paper, we show that cube pruning is essentially equivalent to A* search on a specific search space with specific heuristics. We use this insight to develop faster and exact variants of cube pruning.

1 Introduction

In recent years, an intense research focus on machine translation (MT) has raised the quality of MT systems to the degree that they are now viable for a variety of real-world applications. Because of this, the research community has turned its attention to a major drawback of such systems: they are still quite slow. Recent years have seen a flurry of innovative techniques designed to tackle this problem. These include cube pruning (Chiang, 2007), cube growing (Huang and Chiang, 2007), early pruning (Moore and Quirk, 2007), closing spans (Roark and Hollingshead, 2008; Roark and Hollingshead, 2009), coarse-to-fine methods (Petrov et al., 2008), pervasive laziness (Pust and Knight, 2009), and many more.

This massive interest in speed is bringing rapid progress to the field, but it comes with a certain amount of baggage. Each technique brings its own terminology (from the *cubes* of (Chiang, 2007) to the *lazy lists* of (Pust and Knight, 2009)) into the mix. Often, it is not entirely clear why they work. Many apply only to specialized MT situations. Without a deeper understanding of these methods, it is difficult for the practitioner to combine them and adapt them to new use cases.

In this paper, we attempt to bring some clarity to the situation by taking a closer look at one of these existing methods. Specifically, we cast the popular technique of *cube pruning* (Chiang, 2007) in the well-understood terms of heuristic search (Pearl, 1984). We show that cube pruning is essentially equivalent to A* search on a specific search space with specific heuristics. This simple observation affords a deeper insight into how and why cube pruning works. We show how this insight enables us to easily develop faster and exact variants of cube pruning for tree-to-string transducer-based MT (Galley et al., 2004; Galley et al., 2006; DeNero et al., 2009).

2 Motivating Example

We begin by describing the problem that cube pruning addresses. Consider a synchronous context-free grammar (SCFG) that includes the following rules:

$$\mathrm{A} \rightarrow \langle \mathrm{A}_{\boxed{0}}\ \mathrm{B}_{\boxed{1}},\ \mathrm{A}_{\boxed{0}}\ \mathrm{B}_{\boxed{1}} \rangle \quad (1)$$

$$\mathrm{B} \rightarrow \langle \mathrm{A}_{\boxed{0}}\ \mathrm{B}_{\boxed{1}},\ \mathrm{B}_{\boxed{1}}\ \mathrm{A}_{\boxed{0}} \rangle \quad (2)$$

$$\mathrm{A} \rightarrow \langle \mathrm{B}_{\boxed{0}}\ \mathrm{A}_{\boxed{1}},\ \mathrm{c}\ \mathrm{B}_{\boxed{0}}\ \mathrm{b}\ \mathrm{A}_{\boxed{1}} \rangle \quad (3)$$

$$\mathrm{B} \rightarrow \langle \mathrm{B}_{\boxed{0}}\ \mathrm{A}_{\boxed{1}},\ \mathrm{B}_{\boxed{0}}\ \mathrm{A}_{\boxed{1}} \rangle \quad (4)$$

Figure 1 shows CKY decoding in progress. CKY is a bottom-up algorithm that works by building objects known as *items*, over increasingly larger *spans* of an input sentence (in the context of SCFG decoding, the items represent partial translations of the input sentence). To limit running time, it is common practice to keep only the n "best" items per span (this is known as *beam decoding*). At this point in Figure 1, every span of size 2 or less has already been filled, and now we want to fill span $[2, 5]$ with the n items of lowest cost. Cube pruning addresses the problem of how to compute the n-best items efficiently.

We can be more precise if we introduce some terminology. An SCFG *rule* has the form $\mathrm{X} \rightarrow \langle \sigma, \varphi, \sim \rangle$, where X is a nonterminal (called the *postcondition*), σ, φ are strings that may contain terminals and nonterminals, and $\sim$ is a 1-1 correspondence between equivalent nonterminals of σ and φ.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 62–71,
Singapore, 6-7 August 2009.

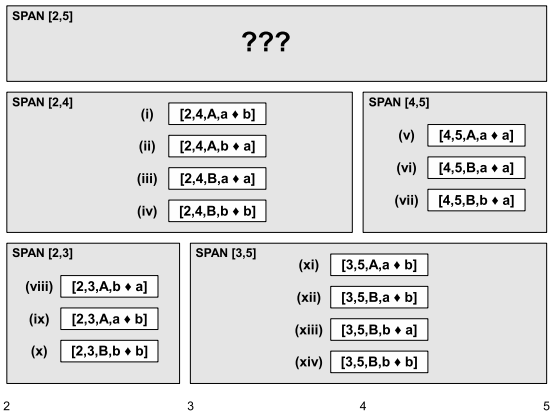


Figure 1: CKY decoding in progress. We want to fill span [2,5] with the lowest cost items.

Usually SCFG rules are represented like the example rules (1)-(4). The subscripts indicate corresponding nonterminals (according to $\sim$). Define the *preconditions* of a rule as the ordered sequence of its nonterminals. For clarity of presentation, we will henceforth restrict our focus to binary rules, i.e. rules of the form: $\mathrm{Z} \rightarrow \langle \mathrm{X}_{\boxed{0}}\ \mathrm{Y}_{\boxed{1}}, \varphi \rangle$. Observe that all the rules of our example are binary rules.

An *item* is a triple that contains a span and two strings. We refer to these strings as the *postcondition* and the *carry*, respectively. The *postcondition* tells us which rules may be applied to the item. The *carry* gives us extra information required to correctly score the item (in SCFG decoding, typically it consists of boundary words for an n-gram language model). [1] To flatten the notation, we will generally represent items as a 4-tuple, e.g. $[2, 4, \mathrm{X}, \mathrm{a} \diamond \mathrm{b}]$.

In CKY, new items are created by applying rules to existing items:

$$\frac{r : \mathrm{Z} \rightarrow \langle \mathrm{X}_{\boxed{0}}\ \mathrm{Y}_{\boxed{1}}, \varphi \rangle \quad [\alpha, \delta, \mathrm{X}, \kappa_1] \quad [\delta, \beta, \mathrm{Y}, \kappa_2]}{[\alpha, \beta, \mathrm{Z}, \mathsf{carry}(r, \kappa_1, \kappa_2)]} \tag{5}$$

In other words, we are allowed to apply a rule r to a pair of items ι_1, ι_2 if the item spans are complementary and $\mathsf{preconditions}(r) = \langle \mathsf{postcondition}(\iota_1), \mathsf{postcondition}(\iota_2) \rangle$. The new item has the same postcondition as the applied rule. We form the carry for the new item through an application-dependent function carry that combines the carries of its subitems (e.g. if the carry is n-gram boundary words, then carry computes the new boundary words). As a shorthand, we introduce the notation $\iota_1 \succcurlyeq r \preccurlyeq \iota_2$ to describe an item created by applying formula (5) to rule r and items ι_1, ι_2.

When we create a new item, it is scored using the following formula: [2]

$$\begin{aligned} \mathsf{cost}(\iota_1 \succcurlyeq r \preccurlyeq \iota_2) \triangleq\ & \mathsf{cost}(r) \\ & + \mathsf{cost}(\iota_1) \\ & + \mathsf{cost}(\iota_2) \\ & + \mathsf{interaction}(r, \kappa_1, \kappa_2) \end{aligned} \tag{6}$$

We assume that each grammar rule r has an associated cost, denoted $\mathsf{cost}(r)$. The *interaction cost*, denoted $\mathsf{interaction}(r, \kappa_1, \kappa_2)$, uses the carry information to compute cost components that cannot be incorporated offline into the rule costs (again, for our purposes, this is a language model score).

Cube pruning addresses the problem of efficiently computing the n items of lowest cost for a given span.

3 Item Generation as Heuristic Search

Refer again to the example in Figure 1. We want to fill span [2,5]. There are 26 distinct ways to apply formula (5), which result in 10 unique items. One approach to finding the lowest-cost n items: perform all 26 distinct inferences, compute the cost of the 10 unique items created, then choose the lowest n.

The 26 different ways to form the items can be structured as a search tree. See Figure 2. First we choose the subspans, then the rule preconditions, then the rule, and finally the subitems. Notice that this search space is already quite large, even for such a simple example. In a realistic situation, we are likely to have a search tree with thousands (possibly millions) of nodes, and we may only want to find the best 100 or so goal nodes. To explore this entire search space seems wasteful. Can we do better?

Why not perform heuristic search directly on this search space to find the lowest-cost n items? In order to do this, we just need to add heuristics to the internal nodes of the space.

Before doing so, it will help to elaborate on some of the details of the search tree. Let $\mathsf{rules}(\mathrm{X}, \mathrm{Y})$ be the subset of rules with preconditions $\langle \mathrm{X}, \mathrm{Y} \rangle$, sorted by increasing cost. Similarly,

[1] Note that the carry is a generic concept that can store any kind of non-local scoring information.

[2] Without loss of generality, we assume an additive cost function.

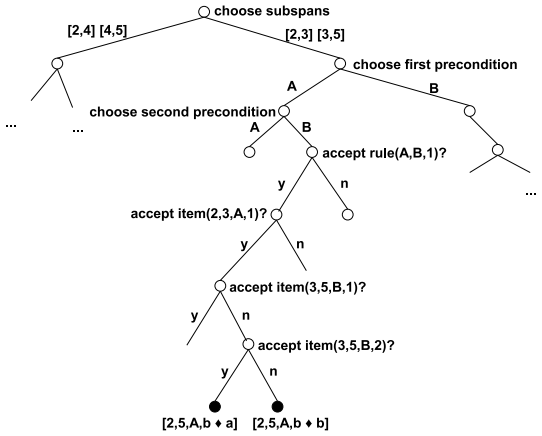


Figure 2: Item creation, structured as a search space. $\mathsf{rule}(\mathrm{X}, \mathrm{Y}, k)$ denotes the k^{th} lowest-cost rule with preconditions $\langle \mathrm{X}, \mathrm{Y} \rangle$. $\mathsf{item}(\alpha, \beta, \mathrm{X}, k)$ denotes the k^{th} lowest-cost item of span $[\alpha, \beta]$ with postcondition X.

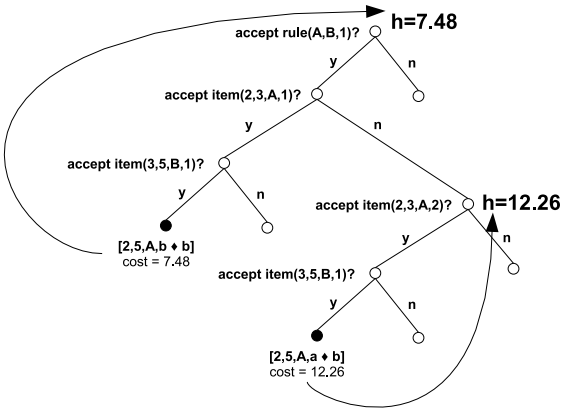


Figure 3: The lookahead heuristic. We set the heuristics for rule and item nodes by looking ahead at the cost of the greedy solution from that point in the search space.

let $\mathsf{items}(\alpha, \beta, \mathrm{X})$ be the subset of items with span $[\alpha, \beta]$ and postcondition X, also sorted by increasing cost. Finally, let $\mathsf{rule}(\mathrm{X}, \mathrm{Y}, k)$ denote the k^{th} rule of $\mathsf{rules}(\mathrm{X}, \mathrm{Y})$ and let $\mathsf{item}(\alpha, \beta, \mathrm{X}, k)$ denote the k^{th} item of $\mathsf{items}(\alpha, \beta, \mathrm{X})$.

A path through the search tree consists of the following sequence of decisions:

1. Set i, j, k to 1.
2. Choose the subspans: $[\alpha, \delta], [\delta, \beta]$.
3. Choose the first precondition X of the rule.
4. Choose the second precondition Y of the rule.
5. While rule not yet accepted and $i < |\mathsf{rules}(\mathrm{X}, \mathrm{Y})|$:
   (a) Choose to accept/reject $\mathsf{rule}(\mathrm{X}, \mathrm{Y}, i)$. If reject, then increment i.
6. While item not yet accepted for subspan $[\alpha, \delta]$ and $j < |\mathsf{items}(\alpha, \delta, \mathrm{X})|$:
   (a) Choose to accept/reject $\mathsf{item}(\alpha, \delta, \mathrm{X}, j)$. If reject, then increment j.
7. While item not yet accepted for subspan $[\delta, \beta]$ and $k < |\mathsf{items}(\delta, \beta, \mathrm{Y})|$:
   (a) Choose to accept/reject $\mathsf{item}(\delta, \beta, \mathrm{Y}, k)$. If reject, then increment k.

Figure 2 shows two complete search paths for our example, terminated by *goal nodes* (in black). Notice that the internal nodes of the search space can be classified by the type of decision they govern. To distinguish between these nodes, we will refer to them as *subspan nodes*, *precondition nodes*, *rule nodes*, and *item nodes*.

We can now proceed to attach heuristics to the nodes and run a heuristic search protocol, say A*, on this search space. For subspan and precondition nodes, we attach trivial uninformative heuristics, i.e. $h = -\infty$. For goal nodes, the heuristic is the actual cost of the item they represent. For rule and item nodes, we will use a simple type of heuristic, often referred to in the literature as a *lookahead heuristic*. Since the rule nodes and item nodes are ordered, respectively, by rule and item cost, it is possible to "look ahead" at a greedy solution from any of those nodes. See Figure 3. This greedy solution is reached by choosing to accept every decision presented until we hit a goal node.

If these heuristics were admissible (i.e. lower bounds on the cost of the best reachable goal node), this would enable us to exactly generate the n-best items without exhausting the search space (assuming the heuristics are strong enough for A* to do some pruning). Here, the lookahead heuristics are clearly not admissible, however the hope is that A* will generate n "good" items, and that the time savings will be worth sacrificing exactness for.

4 Cube Pruning as Heuristic Search

In this section, we will compare cube pruning with our A* search protocol, by tracing through their respective behaviors on the simple example of Figure 1.

4.1 Phase 1: Initialization

To fill span $[\alpha, \beta]$, cube pruning (CP) begins by constructing a *cube* for each tuple of the form:

$$\langle [\alpha, \delta], [\delta, \beta],\ \mathrm{X}\ ,\ \mathrm{Y} \rangle$$

where X and Y are nonterminals. A cube consists of three axes: $\mathsf{rules}(\mathrm{X}, \mathrm{Y})$ and $\mathsf{items}(\alpha, \delta, \mathrm{X})$ and $\mathsf{items}(\delta, \beta, \mathrm{Y})$. Figure 4(left) shows the nontrivial cubes for our example scenario.

Contrast this with A*, which begins by adding the root node of our search space to an empty heap (ordered by heuristic cost). It proceeds to repeatedly pop the lowest-cost node from the heap, then add its children to the heap (we refer to this operation as *visiting* the node). Note that before A* ever visits a rule node, it will have visited every subspan and precondition node (because they all have cost $h = -\infty$). Figure 4(right) shows the state of A* at this point in the search. We assume that we do not generate dead-end nodes (a simple matter of checking that there exist applicable rules and items for the chosen subspans and preconditions). Observe the correspondence between the cubes and the heap contents at this point in the A* search.

4.2 Phase 2: Seeding the Heap

Cube pruning proceeds by computing the "best" item of each cube $\langle [\alpha, \delta], [\delta, \beta],\ \mathrm{X}\ ,\ \mathrm{Y} \rangle$, i.e.

$$\mathsf{item}(\alpha, \delta, \mathrm{X}, 1) \gtrdot \mathsf{rule}(\mathrm{X}, \mathrm{Y}, 1) \lessdot \mathsf{item}(\delta, \beta, \mathrm{Y}, 1)$$

Because of the interaction cost, there is no guarantee that this will really be the best item of the cube, however it is likely to be a good item because the costs of the individual components are low. These items are added to a heap (to avoid confusion, we will henceforth refer to the two heaps as the *CP heap* and the *A* heap*), and prioritized by their costs.

Consider again the example. CP seeds its heap with the "best" items of the 4 cubes. There is now a direct correspondence between the CP heap and the A* heap. Moreover, the costs associated with the heap elements also correspond. See Figure 5.

4.3 Phase 3: Finding the First Item

Cube pruning now pops the lowest-cost item from the CP heap. This means that CP has decided to keep the item. After doing so, it forms the "one-off" items and pushes those onto the CP heap. See Figure 5(left). The popped item is:

$$\text{item (viii)} \gtrdot \text{rule (1)} \lessdot \text{item (xii)}$$

CP then pushes the following one-off successors onto the CP heap:

$$\text{item (viii)} \gtrdot \text{rule (2)} \lessdot \text{item (xii)}$$
$$\text{item (ix)} \gtrdot \text{rule (1)} \lessdot \text{item (xii)}$$
$$\text{item (viii)} \gtrdot \text{rule (1)} \lessdot \text{item (xiii)}$$

Contrast this with A*, which pops the lowest-cost *search node* from the A* heap. Here we need to assume that our A* protocol differs slightly from standard A*. Specifically, it will practice *node-tying*, meaning that when it visits a rule node or an item node, then it also (atomically) visits all nodes on the path to its lookahead goal node. See Figure 5(right). Observe that all of these nodes have the same heuristic cost, thus standard A* is *likely* to visit these nodes in succession without the need to enforce node-tying, but it would not be guaranteed (because the heuristics are not admissible). A* keeps the goal node it finds and adds the successors to the heap, scored with their lookahead heuristics. Again, note the direct correspondence between what CP and A* keep, and what they add to their respective heaps.

4.4 Phase 4: Finding Subsequent Items

Cube pruning and A* continue to repeat Phase 3 until k unique items have been kept. While we could continue to trace through the example, by now it should be clear: cube pruning and our A* protocol with node-tying are doing the same thing at each step. In fact, they are *exactly the same algorithm*. We do not present a formal proof here; this statement should be regarded as confident conjecture.

The node-tying turns out to be an unnecessary artifact. In our early experiments, we discovered that node-tying has no impact on speed or quality. Hence, for the remainder of the paper, we view cube pruning in very simple terms: as nothing more than standard A* search on the search space of Section 3.

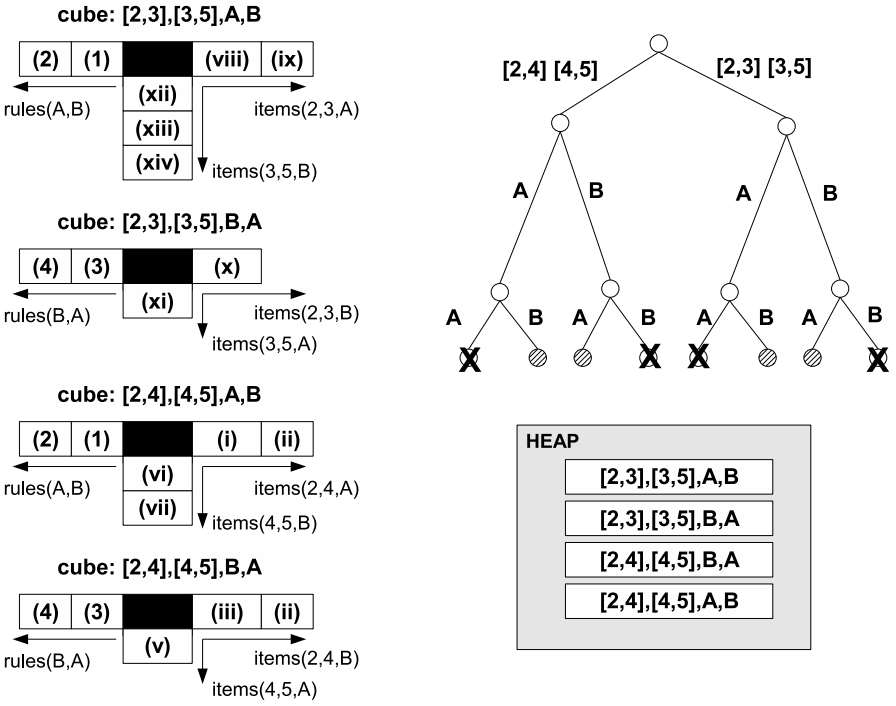


Figure 4: (left) Cube formation for our example. (right) The A* protocol, after all subspan and precondition nodes have been visited. Notice the correspondence between the cubes and the A* heap contents.

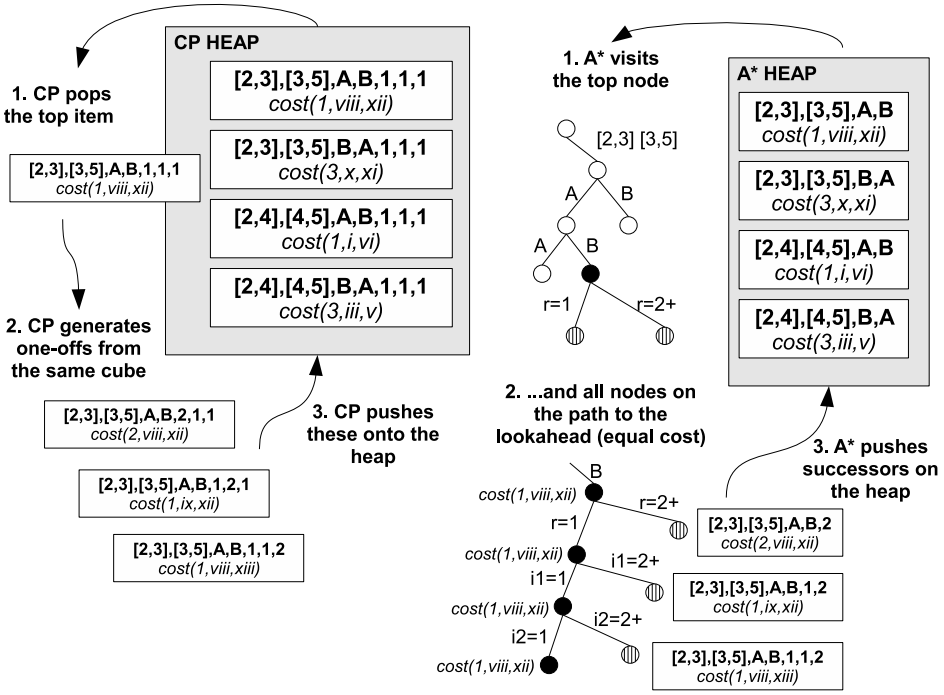


Figure 5: (left) One step of cube pruning. (right) One step of the A* protocol. In this figure, $\mathsf{cost}(r, \iota_1, \iota_2) \triangleq \mathsf{cost}(\iota_1 \succ r \prec \iota_2)$.

5 Augmented Cube Pruning

Viewed in this light, the idiosyncracies of cube pruning begin to reveal themselves. On the one hand, rule and item nodes are associated with strong but inadmissible heuristics (the short explanation for why cube pruning is an inexact algorithm). On the other hand, subspan and precondition nodes are associated with weak trivial heuristics. This should be regarded neither as a surprise nor a criticism, considering cube pruning's origins in hierarchical phrase-based MT models (Chiang, 2007), which have only a small number of distinct nonterminals.

But the situation is much different in tree-to-string transducer-based MT (Galley et al., 2004; Galley et al., 2006; DeNero et al., 2009). Transducer-based MT relies on SCFGs with large nonterminal sets. Binarizing the grammars (Zhang et al., 2006) further increases the size of these sets, due to the introduction of virtual nonterminals.

A key benefit of the heuristic search viewpoint is that it is well positioned to take advantage of such insights into the structure of a particular decoding problem. In the case of transducer-based MT, the large set of preconditions encourages us to introduce a nontrivial heuristic for the precondition nodes. The inclusion of these heuristics into the CP search will enable A* to eliminate certain preconditions from consideration, giving us a speedup. For this reason we call this strategy *augmented cube pruning*.

5.1 Heuristics on preconditions

Recall that the total cost of a goal node is given by Equation (6), which has four terms. We will form the heuristic for a precondition node by creating a separate heuristic for each of the four terms and using the sum as the overall heuristic.

To describe these heuristics, we will make intuitive use of the wildcard operator $*$ to extend our existing notation. For instance, $\mathsf{items}(\alpha, \beta, *)$ will denote the union of $\mathsf{items}(\alpha, \beta, \mathrm{X})$ over all possible X, sorted by cost.

We associate the heuristic $h(\delta, \mathrm{X}, \mathrm{Y})$ with the search node reached by choosing subspans $[\alpha, \delta]$, $[\delta, \beta]$, precondition X (for span $[\alpha, \delta]$), and precondition Y (for span $[\delta, \beta]$). The heuristic is the sum of four terms, mirroring Equation (6):

$$\begin{aligned} h(\delta, \mathrm{X}, \mathrm{Y}) &= \mathsf{cost}(\mathsf{rule}(\mathrm{X}, \mathrm{Y}, 1)) \\ &+ \mathsf{cost}(\mathsf{item}(\alpha, \delta, \mathrm{X}, 1)) \\ &+ \mathsf{cost}(\mathsf{item}(\delta, \beta, \mathrm{Y}, 1)) \\ &+ \mathsf{ih}(\delta, \mathrm{X}, \mathrm{Y}) \end{aligned}$$

The first three terms are admissible because each is simply the minimum possible cost of some choice remaining to be made. To construct the *interaction heuristic* $\mathsf{ih}(\delta, \mathrm{X}, \mathrm{Y})$, consider that in a translation model with an integrated n-gram language model, the interaction cost $\mathsf{interaction}(r, \kappa_1, \kappa_2)$ is computed by adding the language model costs of any new complete n-grams that are created by combining the carries (boundary words) with each other and with the lexical items on the rule's target side, taking into account any reordering that the rule may perform. We construct a backoff-style estimate of these new n-grams by looking at $\mathsf{item}(\alpha, \delta, \mathrm{X}, 1) = [\alpha, \delta, \mathrm{X}, \kappa_1]$, $\mathsf{item}(\delta, \beta, \mathrm{Y}, 1) = [\delta, \beta, \mathrm{Y}, \kappa_2]$, and $\mathsf{rule}(\mathrm{X}, \mathrm{Y}, 1)$. We set $\mathsf{ih}(\delta, \mathrm{X}, \mathrm{Y})$ to be a linear combination of the backoff n-grams of the carries κ_1 and κ_2, as well as any n-grams introduced by the rule. For instance, if

$$\begin{aligned} \kappa_1 &= \mathrm{a\ b} \diamond \mathrm{c\ d} \\ \kappa_2 &= \mathrm{e\ f} \diamond \mathrm{g\ h} \\ \mathsf{rule}(\mathrm{X}, \mathrm{Y}, 1) &= \mathrm{Z} \rightarrow \langle \mathrm{X}_{\boxed{0}}\ \mathrm{Y}_{\boxed{1}},\ \mathrm{X}_{\boxed{0}}\ \mathrm{g\ h\ i}\ \mathrm{Y}_{\boxed{1}} \rangle \end{aligned}$$

then

$$\begin{aligned} \mathsf{ih}(\delta, \mathrm{X}, \mathrm{Y}) &= \gamma_1 \cdot \mathsf{LM}(\mathrm{a}) + \gamma_2 \cdot \mathsf{LM}(\mathrm{a\ b}) \\ &+ \gamma_1 \cdot \mathsf{LM}(\mathrm{e}) + \gamma_2 \cdot \mathsf{LM}(\mathrm{e\ f}) \\ &+ \gamma_1 \cdot \mathsf{LM}(\mathrm{g}) + \gamma_2 \cdot \mathsf{LM}(\mathrm{g\ h}) \\ &+ \gamma_3 \cdot \mathsf{LM}(\mathrm{g\ h\ i}) \end{aligned}$$

The coefficients of the combination are free parameters that we can tune to trade off between more pruning and more admissability. Setting the coefficients to zero gives perfect admissibility but is also weak.

The heuristic for the first precondition node is computed similarly:

$$\begin{aligned} h(\delta, \mathrm{X}, *) &= \mathsf{cost}(\mathsf{rule}(\mathrm{X}, *, 1)) \\ &+ \mathsf{cost}(\mathsf{item}(\alpha, \delta, \mathrm{X}, 1)) \\ &+ \mathsf{cost}(\mathsf{item}(\delta, \beta, *, 1)) \\ &+ \mathsf{ih}(\delta, \mathrm{X}, *) \end{aligned}$$

Standard CP			Augmented CP		
nodes (k)	BLEU	time	nodes (k)	BLEU	time
80	34.9	2.5	52	34.7	1.9
148	36.1	3.9	92	35.9	2.4
345	37.2	7.9	200	37.3	5.4
520	37.7	13.4	302	37.7	8.5
725	38.2	17.1	407	38.0	10.7
1092	38.3	27.1	619	38.2	16.3
1812	38.6	45.9	1064	38.5	27.7

Table 1: Results of standard and augmented cube pruning. The number of (thousands of) search nodes visited is given along with BLEU and average time to decode one sentence, in seconds.

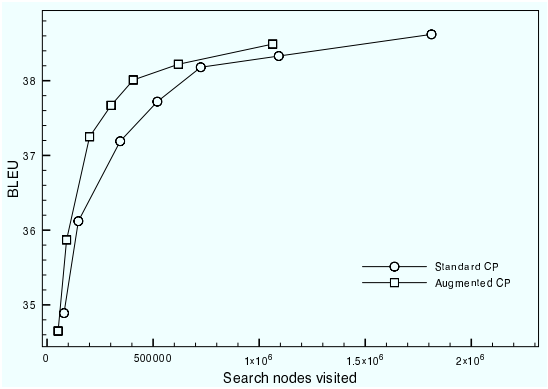


Figure 6: Nodes visited by standard and augmented cube pruning.

We also apply analogous heuristics to the subspan nodes.

5.2 Experimental setup

We evaluated all of the algorithms in this paper on a syntax-based Arabic-English translation system based on (Galley et al., 2006), with rules extracted from 200 million words of parallel data from NIST 2008 and GALE data collections, and with a 4-gram language model trained on 1 billion words of monolingual English data from the LDC Gigaword corpus. We evaluated the system's performance on the NIST 2008 test corpus, which consists of 1357 Arabic sentences from a mixture of newswire and web domains, with four English reference translations. We report BLEU scores (Papineni et al., 2002) on untokenized, recapitalized output.

5.3 Results for Augmented Cube Pruning

The results for augmented cube pruning are compared against cube pruning in Table 1. The data

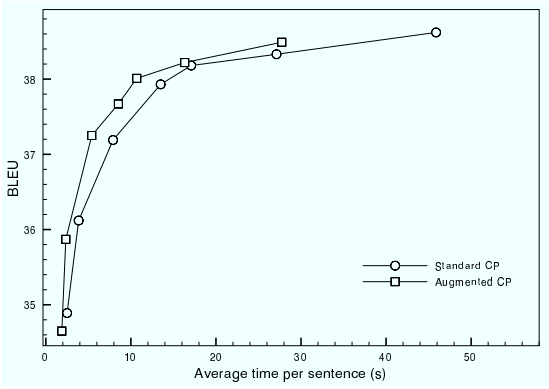


Figure 7: Time spent by standard and augmented cube pruning, average seconds per sentence.

	Standard CP	Augmented CP
subspan	12936	12792
precondition	851458	379954
rule	33734	33331
item	119703	118889
goal	74618	74159
TOTAL	1092449	619125
BLEU	38.33	38.22

Table 2: Breakdown of visited search nodes by type (for a fixed beam size).

from that table are also plotted in Figure 6 and Figure 7. Each line gives the number of nodes visited by the heuristic search, the average time to decode one sentence, and the BLEU of the output. The number of items kept by each span (the beam) is increased in each subsequent line of the table to indicate how the two algorithms differ at various beam sizes. This also gives a more complete picture of the speed/BLEU tradeoff offered by each algorithm. Because the two algorithms make the same sorts of lookahead computations with the same implementation, they can be most directly compared by examining the number of visited nodes. Augmenting cube pruning with admissible heuristics on the precondition nodes leads to a substantial decrease in visited nodes, by 35-44%. The reduction in nodes converges to a consistent 40% as the beam increases. The BLEU with augmented cube pruning drops by an average of 0.1 compared to standard cube pruning. This is due to the additional inadmissibility of the interaction heuristic.

To see in more detail how the heuristics affect the search, we give in Table 2 the number of nodes of each type visited by both variants for one beam

size. The precondition heuristic enables A* to prune more than half the precondition nodes.

6 Exact Cube Pruning

Common wisdom is that the speed of cube pruning more than compensates for its inexactness (recall that this inexactness is due to the fact that it uses A* search with inadmissible heuristics). Especially when we move into transducer-based MT, the search space becomes so large that brute-force item generation is much too slow to be practical. Still, within the heuristic search framework we may ask the question: is it possible to apply strictly admissible heuristics to the cube pruning search space, and in so doing, create a version of cube pruning that is both fast *and* exact, one that finds the n *best* items for each span and not just n *good* items? One might not expect such a technique to outperform cube pruning in practice, but for a given use case, it would give us a relatively fast way of assessing the BLEU drop incurred by the inexactness of cube pruning.

Recall again that the total cost of a goal node is given by Equation (6), which has four terms. It is easy enough to devise strong lower bounds for the first three of these terms by extending the reasoning of Section 5. Table 3 shows these heuristics. The major challenge is to devise an effective lower bound on the fourth term of the cost function, the interaction heuristic, which in our case is the incremental language model cost.

We take advantage of the following observations:

1. *In a given span, many boundary word patterns are repeated.* In other words, for a particular span $[\alpha, \beta]$ and carry κ, we often see many items of the form $[\alpha, \beta, \mathrm{X}, \kappa]$, where the only difference is the postcondition X.

2. *Most rules do not introduce lexical items.* In other words, most of the grammar rules have the form $\mathrm{Z} \rightarrow \langle \mathrm{X}_0\ \mathrm{Y}_1,\ \mathrm{X}_0\ \mathrm{Y}_1 \rangle$ (concatenation rules) or $\mathrm{Z} \rightarrow \langle \mathrm{X}_0\ \mathrm{Y}_1,\ \mathrm{Y}_1\ \mathrm{X}_0 \rangle$ (inversion rules).

The idea is simple. We split the search into three searches: one for concatenation rules, one for inversion rules, and one for lexical rules. Each search finds the n–best items that can be created using its respective set of rules. We then take these $3n$ items and keep the best n.

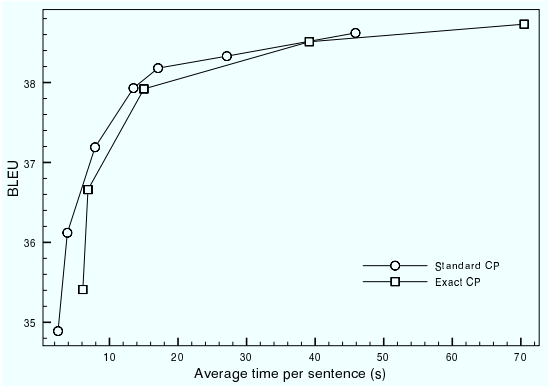


Figure 8: Time spent by standard and exact cube pruning, average seconds per sentence.

Doing this split enables us to precompute a strong and admissible heuristic on the interaction cost. Namely, for a given span $[\alpha, \beta]$, we precompute $\mathsf{ih}_{adm}(\delta, \mathrm{X}, \mathrm{Y})$, which is the best LM cost of combining carries from $\mathsf{items}(\alpha, \delta, \mathrm{X})$ and $\mathsf{items}(\delta, \beta, \mathrm{Y})$. Notice that this statistic is only straightforward to compute once we can assume that the rules are concatenation rules or inversion rules. For the lexical rules, we set $\mathsf{ih}_{adm}(\delta, \mathrm{X}, \mathrm{Y}) = 0$, an admissible but weak heuristic that we can fortunately get away with because of the small number of lexical rules.

6.1 Results for Exact Cube Pruning

Computing the $\mathsf{ih}_{adm}(\delta, \mathrm{X}, \mathrm{Y})$ heuristic is not cheap. To be fair, we first compare exact CP to standard CP in terms of overall running time, including the computational cost of this overhead. We plot this comparison in Figure 8. Surprisingly, the time/quality tradeoff of exact CP is extremely similar to standard CP, suggesting that exact cube pruning is actually a practical alternative to standard CP, and not just of theoretical value. We found that the BLEU loss of standard cube pruning at moderate beam sizes was between 0.4 and 0.6.

Another surprise comes when we contrast the number of visited search nodes of exact CP and standard CP. See Figure 9. While we initially expected that exact CP must visit fewer nodes to make up for the computational overhead of its expensive heuristics, this did not turn out to be the case, suggesting that the computational cost of standard CP's lookahead heuristics is just as expensive as the precomputation of $\mathsf{ih}_{adm}(\delta, \mathrm{X}, \mathrm{Y})$.

	heuristic components					
	subspan $h(\delta)$	precondition1 $h(\delta, \mathrm{X})$	precondition2 $h(\delta, \mathrm{X}, \mathrm{Y})$	rule $h(\delta, \mathrm{X}, \mathrm{Y}, i)$	item1 $h(\delta, \mathrm{X}, \mathrm{Y}, i, j)$	item2 $h(\delta, \mathrm{X}, \mathrm{Y}, i, j, k)$
r	$\mathsf{rule}(*, *, 1)$	$\mathsf{rule}(\mathrm{X}, *, 1)$	$\mathsf{rule}(\mathrm{X}, \mathrm{Y}, 1)$	$\mathsf{rule}(\mathrm{X}, \mathrm{Y}, i)$	$\mathsf{rule}(\mathrm{X}, \mathrm{Y}, i)$	$\mathsf{rule}(\mathrm{X}, \mathrm{Y}, i)$
ι_1	$\mathsf{item}(\alpha, \delta, *, 1)$	$\mathsf{item}(\alpha, \delta, \mathrm{X}, 1)$	$\mathsf{item}(\alpha, \delta, \mathrm{X}, 1)$	$\mathsf{item}(\alpha, \delta, \mathrm{X}, 1)$	$\mathsf{item}(\alpha, \delta, \mathrm{X}, j)$	$\mathsf{item}(\alpha, \delta, \mathrm{X}, j)$
ι_2	$\mathsf{item}(\delta, \beta, *, 1)$	$\mathsf{item}(\delta, \beta, *, 1)$	$\mathsf{item}(\delta, \beta, \mathrm{Y}, 1)$	$\mathsf{item}(\delta, \beta, \mathrm{Y}, 1)$	$\mathsf{item}(\delta, \beta, \mathrm{Y}, 1)$	$\mathsf{item}(\delta, \beta, \mathrm{Y}, k)$
ih	$\mathsf{ih}_{adm}(\delta, *, *)$	$\mathsf{ih}_{adm}(\delta, \mathrm{X}, *)$	$\mathsf{ih}_{adm}(\delta, \mathrm{X}, \mathrm{Y})$	$\mathsf{ih}_{adm}(\delta, \mathrm{X}, \mathrm{Y})$	$\mathsf{ih}_{adm}(\delta, \mathrm{X}, \mathrm{Y})$	$\mathsf{ih}_{adm}(\delta, \mathrm{X}, \mathrm{Y})$

Table 3: Admissible heuristics for exact CP. We attach heuristic $h(\delta, \mathrm{X}, \mathrm{Y}, i, j, k)$ to the search node reached by choosing subspans $[\alpha, \delta], [\delta, \beta]$, preconditions X and Y, the i^{th} rule of $\mathsf{rules}(\mathrm{X}, \mathrm{Y})$, the j^{th} item of $\mathsf{item}(\alpha, \delta, \mathrm{X})$, and the k^{th} item of $\mathsf{item}(\delta, \beta, \mathrm{Y})$. To form the heuristic for a particular type of search node (column), compute the following: $\mathsf{cost}(r) + \mathsf{cost}(\iota_1) + \mathsf{cost}(\iota_2) + ih$

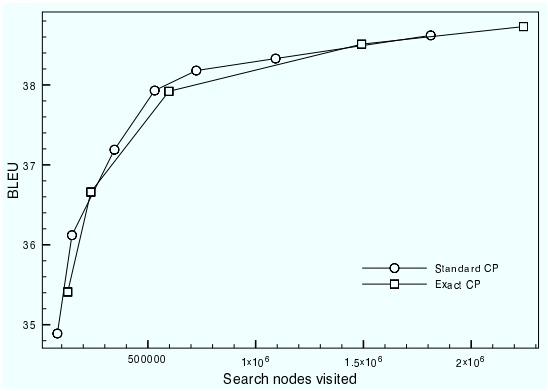


Figure 9: Nodes visited by standard and exact cube pruning.

7 Implications

This paper's core idea is the utility of framing CKY item generation as a heuristic search problem. Once we recognize cube pruning as nothing more than A* on a particular search space with particular heuristics, this deeper understanding makes it easy to create faster and exact variants for other use cases (in this paper, we focus on tree-to-string transducer-based MT). Depending on one's own particular use case, a variety of possibilities may present themselves:

1. *What if we try different heuristics?* In this paper, we do some preliminary inquiry into this question, but it should be clear that our minor changes are just the tip of the iceberg. One can easily imagine clever and creative heuristics that outperform the simple ones we have proposed here.

2. *What if we try a different search space?* Why are we using this particular search space? Perhaps a different one, one that makes decisions in a different order, would be more effective.

3. *What if we try a different search algorithm?* A* has nice guarantees (Dechter and Pearl, 1985), but it is space-consumptive and it is not anytime. For a use case where we would like a finer-grained speed/quality tradeoff, it might be useful to consider an anytime search algorithm, like depth-first branch-and-bound (Zhang and Korf, 1995).

By working towards a deeper and unifying understanding of the smorgasbord of current MT speed-up techniques, our hope is to facilitate the task of implementing such methods, combining them effectively, and adapting them to new use cases.

Acknowledgments

We would like to thank Abdessamad Echihabi, Kevin Knight, Daniel Marcu, Dragos Munteanu, Ion Muslea, Radu Soricut, Wei Wang, and the anonymous reviewers for helpful comments and advice. Thanks also to David Chiang for the use of his LaTeX macros. This work was supported in part by CCS grant 2008-1245117-000.

References

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Rina Dechter and Judea Pearl. 1985. Generalized best-first search strategies and the optimality of a*. *Journal of the ACM*, 32(3):505–536.

John DeNero, Mohit Bansal, Adam Pauls, and Dan Klein. 2009. Efficient parsing for transducer grammars. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*.

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

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

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL*.

Robert C. Moore and Chris Quirk. 2007. Faster beam-search decoding for phrasal statistical machine translation. In *Proceedings of MT Summit XI*.

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

Judea Pearl. 1984. *Heuristics*. Addison-Wesley.

Slav Petrov, Aria Haghighi, and Dan Klein. 2008. Coarse-to-fine syntactic machine translation using language projections. In *Proceedings of EMNLP*.

Michael Pust and Kevin Knight. 2009. Faster mt decoding through pervasive laziness. In *Proceedings of NAACL*.

Brian Roark and Kristy Hollingshead. 2008. Classifying chart cells for quadratic complexity context-free inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 745–752.

Brian Roark and Kristy Hollingshead. 2009. Linear complexity context-free parsing pipelines via chart constraints. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 647–655, Boulder, Colorado, June. Association for Computational Linguistics.

Weixiong Zhang and Richard E. Korf. 1995. Performance of linear-space search algorithms. *Artificial Intelligence*, 79(2):241–292.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 256–263.

Effective Use of Linguistic and Contextual Information for Statistical Machine Translation

Libin Shen and **Jinxi Xu** and **Bing Zhang** and
Spyros Matsoukas and **Ralph Weischedel**
BBN Technologies
Cambridge, MA 02138, USA
{lshen,jxu,bzhang,smatsouk,weisched}@bbn.com

Abstract

Current methods of using lexical features in machine translation have difficulty in scaling up to realistic MT tasks due to a prohibitively large number of parameters involved. In this paper, we propose methods of using new linguistic and contextual features that do not suffer from this problem and apply them in a state-of-the-art hierarchical MT system. The features used in this work are non-terminal labels, non-terminal length distribution, source string context and source dependency LM scores. The effectiveness of our techniques is demonstrated by significant improvements over a strong baseline. On Arabic-to-English translation, improvements in lower-cased BLEU are 2.0 on NIST MT06 and 1.7 on MT08 newswire data on decoding output. On Chinese-to-English translation, the improvements are 1.0 on MT06 and 0.8 on MT08 newswire data.

1 Introduction

Linguistic and context features, especially sparse lexical features, have been widely used in recent machine translation (MT) research. Unfortunately, existing methods of using such features are not ideal for large-scale, practical translation tasks.

In this paper, we will propose several probabilistic models to effectively exploit linguistic and contextual information for MT decoding, and these new features do not suffer from the scalability problem. Our new models are tested on NIST MT06 and MT08 data, and they provide significant improvement over a strong baseline system.

1.1 Previous Work

The ideas of using labels, length preference and source side context in MT decoding were explored previously. Broadly speaking, two approaches were commonly used in existing work.

One is to use a stochastic gradient descent (SGD) or Perceptron like online learning algorithm to optimize the weights of these features directly for MT (Shen et al., 2004; Liang et al., 2006; Tillmann and Zhang, 2006). This method is very attractive, since it opens the door to rich lexical features. However, in order to robustly optimize the feature weights, one has to use a substantially large development set, which results in significantly slower tuning. Alternatively, one needs to carefully select a development set that simulates the test set to reduce the risk of over-fitting, which however is not always realistic for practical use.

A remedy is to aggressively limit the feature space, e.g. to syntactic labels or a small fraction of the bi-lingual features available, as in (Chiang et al., 2008; Chiang et al., 2009), but that reduces the benefit of lexical features. A possible generic solution is to cluster the lexical features in some way. However, how to make it work on such a large space of bi-lingual features is still an open question.

The other approach is to estimate a single score or likelihood of a translation with rich features, for example, with the maximum entropy (MaxEnt) method as in (Carpuat and Wu, 2007; Ittycheriah and Roukos, 2007; He et al., 2008). This method avoids the over-fitting problem, at the expense of losing the benefit of discriminative training of rich features directly for MT. However, the feature space problem still exists in these published models.

He et al. (2008) extended the WSD-like approached proposed in (Carpuat and Wu, 2007) to hierarchical decoders. In (He et al., 2008), lexical

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 72–80,
Singapore, 6-7 August 2009.

features were limited on each single side due to the feature space problem. In order to further reduce the complexity of MaxEnt training, they "trained a MaxEnt model for each ambiguous hierarchical LHS" (left-hand side or source side) of translation rules. Different target sides were treated as possible labels. Therefore, the sample sets of each individual MaxEnt model were very small, while the number of features could easily exceed the number of samples. Furthermore, optimizing individual MaxEnt models in this way does not lead to global maximum. In addition, MaxEnt models trained on small sets are unstable.

The MaxEnt model in (Ittycheriah and Roukos, 2007) was optimized globally, so that it could better employ the distribution of the training data. However, one has to filter the training data according to the test data to get competitive performance with this model [1]. In addition, the filtering method causes some practical issues. First, such methods are not suitable for real MT tasks, especially for applications with streamed input, since the model has to be retrained with each new input sentence or document and training is slow. Furthermore, the model is ill-posed. The translation of a source sentence depends on other source sentences in the same batch with which the MaxEnt model is trained. If we add one more sentence to the batch, translations of other sentences may become different due to the change of the MaxEnt model.

To sum up, the existing models of employing rich bi-lingual lexical information in MT are imperfect. Many of them are not ideal for practical translation tasks.

1.2 Our Approach

As for our approach, we mainly use simple probabilistic models, i.e. Gaussian and n-gram models, which are more robust and suitable for large-scale training of real data, as manifested in state-of-the-art systems of speech recognition. The unique contribution of our work is to design effective and efficient statistical models to capture useful linguistic and context information for MT decoding. Feature functions defined in this way are robust and ideal for practical translation tasks.

[1] According to footnote 2 of (Ittycheriah and Roukos, 2007), test set adaptation by test set sampling of the training corpus showed an advantage of more than 2 BLEU points over a general system trained on all data.

1.2.1 Features

In this paper, we will introduce four new linguistic and contextual feature functions. Here, we first provide a high-level description of these features. Details of the features are discussed in Section 2.

The first feature is based on non-terminal labels, i.e. POS tags of the head words of target non-terminals in transfer rules. This feature reduces the ambiguity of translation rules. The other benefit is that POS tags help to weed out bad target side tree structures, as an enhancement to the target dependency language model.

The second feature is based on the length distribution of non-terminals. In English as well as in other languages, the same deep structure can be represented in different syntactic structures depending on the complexity of its constituents. We model such preferences by associating each non-terminal of a transfer rule with a probability distribution over its length. Similar ideas were explored in (He et al., 2008). However their length features only provided insignificant improvement of 0.1 BLEU point. A crucial difference of our approach is how the length preference is modeled. We approximate the length distribution of non-terminals with a smoothed Gaussian, which is more robust and gives rise to much larger improvement consistently.

The third feature utilizes source side context information, i.e. the neighboring words of an input span, to influence the selection of the target translation for a span. While the use of context information has been explored in MT, e.g. (Carpuat and Wu, 2007) and (He et al., 2008), the specific technique we used by means of a context language model is rather different. Our model is trained on the whole training data, and it is not limited by the constraint of MaxEnt training.

The fourth feature exploits structural information on the source side. Specifically, the decoder simultaneously generates both the source and target side dependency trees, and employs two dependency LMs, one for the source and the other for the target, for scoring translation hypotheses. Our intuition is that the likelihood of source structures provides another piece of evidence about the plausibility of a translation hypothesis and as such would help weed out bad ones.

1.2.2 Baseline System and Experimental Setup

We take BBN's *HierDec*, a string-to-dependency decoder as described in (Shen et al., 2008), as our baseline for the following two reasons:

- It provides a strong baseline, which ensures the validity of the improvement we would obtain. The baseline model used in this paper showed state-of-the-art performance at NIST 2008 MT evaluation.
- The baseline algorithm can be easily extended to incorporate the features proposed in this paper. The use of source dependency structures is a natural extension of the string-to-tree model to a tree-to-tree model.

To ensure the generality of our results, we tested the features on two rather different language pairs, Arabic-to-English and Chinese-to-English, using two metrics, IBM BLEU (Papineni et al., 2001) and TER (Snover et al., 2006). Our experiments show that each of the first three features: non-terminal labels, length distribution and source side context, improves MT performance. Surprisingly, the source dependency feature does not produce an improvement.

2 Linguistic and Context Features

2.1 Non-terminal Labels

In the original string-to-dependency model (Shen et al., 2008), a translation rule is composed of a string of words and non-terminals on the source side and a well-formed dependency structure on the target side. A well-formed dependency structure could be either a single-rooted dependency tree or a set of sibling trees. As in the Hiero system (Chiang, 2007), there is only one non-terminal X in the string-to-dependency model. Any sub dependency structure can be used to replace a non-terminal in a rule.

For example, we have a source sentence in Chinese as follows.

- jiantao zhuyao baohan liang fangmian

The literal translation for individual words is

- 'review' 'mainly' 'to consist of' 'two' 'part'

The reference translation is

- the review mainly consists of two parts

A single source word can be translated into many English words. For example, *jiantao* can be translated into *a review, the review, reviews, the reviews, reviewing, reviewed*, etc. Suppose we have source-string-to-target-dependency translation rules as shown in Figure 1. Since there is no constraint on substitution, any translation for *jiantao* could replace the X-1 slot.

One way to alleviate this problem is to limit the search space by using a label system. We could assign a label to each non-terminal on the target side of the rules. Furthermore, we could assign a label to the whole target dependency structure, as shown in Figure 2. In decoding, each target dependency sub-structure would be associated with a label. Whenever substitution happens, we would check whether the label of the sub-structure and the label of the slot are the same. Substitutions with unmatched labels would be prohibited.

In practice, we use a soft constraint by penalizing substitutions with unmatched labels. We introduce a new feature: the number of times substitutions with unmatched labels appear in the derivation of a translation hypothesis.

Obviously, to implement this feature we need to associate a label with each non-terminal in the target side of a translation rule. The labels are generated during rule extraction. When we create a rule from a training example, we replace a subtree or dependency structure with a non-terminal and associate it with the POS tag of the head word if the non-terminal corresponds to a single-rooted tree on the target side. Otherwise, it is assigned the generic label X. (In decoding, all substitutions of X are considered unmatched ones and incur a penalty.)

2.2 Length Distribution

In English, the length of a phrase may determine the syntactic structure of a sentence. For example, possessive relations can be represented either as "A's B" or "B of A". The former is preferred if A is a short phrase (e.g. "the boy's mother") while the latter is preferred if A is a complex structure (e.g. "the mother of the boy who is sick").

Our solution is to build a model of length preference for each non-terminal in each translation rule. To address data sparseness, we assume the length distribution of each non-terminal in a transfer rule is a Gaussian, whose mean and variance can be estimated from the training data. In rule extrac-

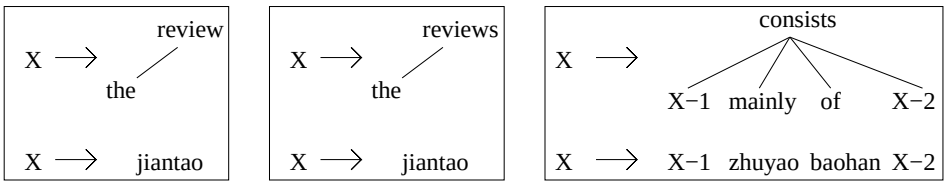


Figure 1: Translation rules with one label X

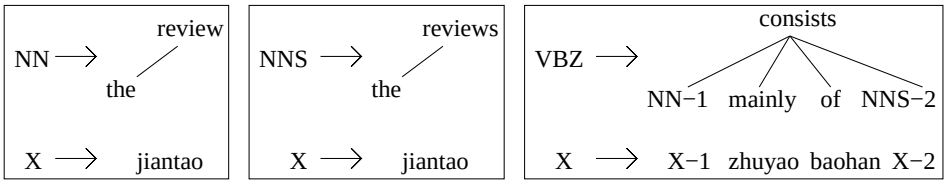


Figure 2: Translation rules with multiple labels

tion, each time a translation rule is generated from a training example, we can record the length of the source span corresponding to a non-terminal. In the end, we have a frequency histogram for each non-terminal in each translation rule. From the histogram, a Gaussian distribution can be easily computed.

In practice, we do not need to collect the frequency histogram. Since all we need to know are the mean and the variance, it is sufficient to collect the sum of the length and the sum of squared length.

Let r be a translation rule that occurs N_r times in training. Let x be a specific non-terminal in that rule. Let $l(r, x, i)$ denote the length of the source span corresponding to non-terminal x in the i-th occurrence of rule r in training. Then, we can compute the following quantities.

$$m_{r,x} = \frac{1}{N_r} \sum_{i=1}^{N_r} l(r, x, i) \quad (1)$$

$$s_{r,x} = \frac{1}{N_r} \sum_{i=1}^{N_r} l(r, x, i)^2, \quad (2)$$

which can be subsequently used to estimate the mean $\mu_{r,x}$ and variance $\sigma^2_{r,x}$ of x's length distribution in rule r as follows.

$$\mu_{r,x} = m_{r,x} \quad (3)$$

$$\sigma^2_{r,x} = s_{r,x} - m^2_{r,x} \quad (4)$$

Since many of the translation rules have few occurrences in training, smoothing of the above estimates is necessary. A common smoothing method is based on maximum a posteriori (MAP) estimation as in (Gauvain and Lee, 1994).

$$\hat{m}_{r,x} = \frac{N_r}{N_r + \tau} m_{r,x} + \frac{\tau}{N_r + \tau} \tilde{m}_{r,x}$$

$$\hat{s}_{r,x} = \frac{N_r}{N_r + \tau} s_{r,x} + \frac{\tau}{N_r + \tau} \tilde{s}_{r,x},$$

where $\hat{}$ stands for an MAP distribution and $\tilde{}$ represents a prior distribution. $\tilde{m}_{r,x}$ and $\tilde{s}_{r,x}$ can be obtained from a prior Gaussian distribution $\mathcal{N}(\tilde{\mu}_{r,x}, \tilde{\sigma}_{r,x})$ via equations (3) and (4), and τ is a weight of smoothing.

There are many ways to approximate the prior distribution. For example, we can have one prior for all the non-terminals or one for individual non-terminal type. In practice, we assume $\tilde{\mu}_{r,x} = \mu_{r,x}$, and approximate $\tilde{\sigma}_{r,x}$ as $(\sigma^2_{r,x} + s_{r,x})^{\frac{1}{2}}$.

In this way, we do not change the mean, but relax the variance with $s_{r,x}$. We tried different smoothing methods, but the performance did not change much, therefore we kept this simplest setup. We also tried the Poisson distribution, and the performance is similar to Gaussian distribution, which is about 0.1 point lower in BLEU.

When a rule r is applied during decoding, we compute a penalty for each non-terminal x in r according to

$$P(l \mid r, x) = \frac{1}{\sigma_{r,x}\sqrt{2\pi}} e^{-\frac{(l-\mu_{r,x})^2}{2\sigma^2_{r,x}}},$$

where l is length of source span corresponding to x.

Our method to address the problem of length bias in rule selection is very different from the maximum entropy method used in existing studies, e.g. (He et al., 2008).

2.3 Context Language Model

In the baseline string-to-dependency system, the probability a translation rule is selected in decoding does not depend on the sentence context. In reality, translation is highly context dependent. To address this defect, we introduce a new feature, called *context language model*. The motivation of this feature is to exploit surrounding words to influence the selection of the desired transfer rule for a given input span.

To illustrate the problem, we use the same example mentioned in Section 2.1. Suppose the source span for rule selection is *zhuyao baohan*, whose literal translation is *mainly* and *to consist of*. There are many candidate translations for this phrase, for example, *mainly consist of, mainly consists of, mainly including, mainly includes*, etc. The surrounding words can help to decide which translation is more appropriate for *zhuyao baohan*. We compare the following two context-based probabilities:

- $P($ jiantao $|$ mainly consist $)$
- $P($ jiantao $|$ mainly consists $)$

Here, *jiantao* is the source word preceding the source span *zhuyao baohan*.

In the training data, *jiantao* is usually translated into *the review*, third-person singular, then the probability $P($ jiantao $|$ mainly consists $)$ will be higher than $P($ jiantao $|$ mainly consist $)$, since we have seen more context events like the former in the training data.

Now we introduce context LM formally. Let the source words be $f_1 f_2 .. f_i .. f_j .. f_n$. Suppose source sub-string $f_i .. f_j$ is translated into $e_p .. e_q$. We can define tri-gram probabilities on the left and right sides of the source span:

- left : $P_L(f_{i-1}|e_p, e_{p+1})$
- right : $P_R(f_{j+1}|e_q, e_{q-1})$

In our implementation, the left and right context LMs are estimated from the training data as part of the rule extraction procedure. When we exact a rule, we collect two 3-gram events, one for the left side and the other for the right side.

In decoding, whenever a partial hypothesis is generated, we calculate the context LM scores based on the leftmost two words and the rightmost two words of the hypothesis as well as the source context. The product of the left and right context LM scores is used as a new feature in the scoring function.

Please note that our approach is very different from other approaches to context dependent rule selection such as (Ittycheriah and Roukos, 2007) and (He et al., 2008). Instead of using a large number of fine grained features with weights optimized using the maximum entropy method, we treat context dependency as an ngram LM problem, and it is smoothed with Witten-Bell discounting. The estimation of the context LMs is very efficient and robust.

The benefit is two fold. The estimation of the context LMs is very efficient. It adds only one new weight to the scoring function.

2.4 Source Dependency Language Model

The context LM proposed in the previous section only employs source words immediately before and after the current source span in decoding. To exploit more source context, we use a source side dependency language model as another feature. The motivation is to take advantage of the long distance dependency relations between source words in scoring a translation theory.

We extended string-to-dependency rules in the baseline system to dependency-to-dependency rules. In each dependency-to-dependency rule, we keep record of the source string as well as the source dependency structure. Figure 3 shows examples of dependency-to-dependency rules.

We extended the string-to-dependency decoding algorithm in the baseline to accommodate dependency-to-dependency theories. In decoding, we build both the source and the target dependency structures simultaneously in chart parsing over the source string. Thus, we can compute the source dependency LM score in the same way we compute the target side score, using a procedure described in (Shen et al., 2008).

We introduce two new features for the source side dependency LM as follows, in a way similar to the target side.

- Source dependency LM score
- Discount on ill-formed source dependency structures

The source dependency LM is trained on the source side of the bi-lingual training data with Witten-Bell smoothing. The source dependency LM score represents the likelihood of the source

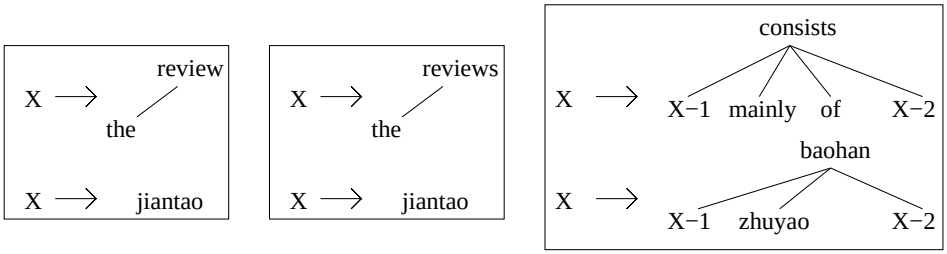


Figure 3: Dependency-to-dependency translation rules

dependency tree generated by the decoder. The source dependency tree with the highest score is the one that is most likely to be generated by the dependency model that created the source side of the training data.

Source dependency trees are composed of fragments embedded in the translation rules. Therefore, a source dependency LM score can be viewed as a measure whether the translation rules are put together in a way similar to the training data. Therefore, a source dependency LM score serves as a feature to represent structural context information that is capable of modeling long-distance relations.

However, unlike source context LMs, the structural context information is used only when two partial dependency structures are combined, while source context LMs work as a look-ahead feature.

3 Experiments

We designed our experiments to show the impact of each feature separately as well as their cumulative impact:

- BASE: baseline string-to-dependency system
- SLM: baseline + source dependency LM
- CLM: baseline + context LM
- LEN: baseline + length distribution
- LBL: baseline + syntactic labels
- LBL+LEN: baseline + syntactic labels + length distribution
- LBL+LEN+CLM: baseline + syntactic labels + length distribution + context LM

All the models were optimized on lower-cased IBM BLEU with Powell's method (Powell, 1964; Brent, 1973) on n-best translations (Ostendorf et al., 1991), but evaluated on both IBM BLEU and TER. The motivation is to detect if an improvement is artificial, i.e., specific to the tuning metric. For both Arabic-to-English and Chinese-to-English MT, we tuned on NIST MT02-05 and tested on MT06 and MT08 newswire sets.

The training data are different from what was usd at MT06 or MT08. Our Arabic-to-English data contain 29M Arabic words and 38M English words from 11 corpora: LDC2004T17, LDC2004T18, LDC2005E46, LDC2006E25, LDC2006G05, LDC2005E85, LDC2006E36, LDC2006E82, LDC2006E95, Sakhr-A2E and Sakhr-E2A. The Chinese-to-English data contain 107M Chinese words and 132M English words from eight corpora: LDC2002E18, LDC2005T06, LDC2005T10, LDC2006E26, LDC2006G05, LDC2002L27, LDC2005T34 and LDC2003E07. They are available under the DARPA GALE program. Traditional 3-gram and 5-gram string LMs were trained on the English side of the parallel data plus the English Gigaword corpus V3.0 in a way described in (Bulyko et al., 2007).

The target dependency LMs were trained on the English side of the parallel training data. For that purpose, we parsed the English side of the parallel data. Two separate models were trained: one for Arabic from the Arabic training data and the other for Chinese from the Chinese training data.

To compute the source dependency LM for Chinese-to-English MT, we parsed the Chinese side of the Chinese-to-English parallel data. Due to the lack of a good Arabic parser compatible with the Sakhr tokenization that we used on the source side, we did not test the source dependency LM for Arabic-to-English MT.

When extracting rules with source dependency structures, we applied the same *well-formedness* constraint on the source side as we did on the target side, using a procedure described by (Shen et al., 2008). Some candidate rules were thrown away due to the source side constraint. On the

	MT06				MT08			
Model	**BLEU**		**TER**		**BLEU**		**TER**	
	lower	**mixed**	**lower**	**mixed**	**lower**	**mixed**	**lower**	**mixed**
Decoding (3-gram LM)								
BASE	48.75	46.74	43.43	45.79	49.58	47.46	42.80	45.08
CLM	49.44	47.36	42.96	45.22	49.73	47.53	42.64	44.92
LEN	49.37	47.28	43.01	45.35	50.29	48.19	42.32	44.45
LBL	49.33	47.07	43.09	45.53	50.46	48.19	42.27	44.57
LBL+LEN	49.91	47.70	42.59	45.17	51.10	48.85	41.88	44.16
LBL+LEN+CLM	50.75	48.51	42.13	44.50	51.24	49.10	41.63	43.80
Rescoring (5-gram LM)								
BASE	51.24	49.23	42.08	44.42	51.23	49.11	42.01	44.15
CLM	51.57	49.54	41.74	43.88	51.44	49.37	41.63	43.74
LEN	52.05	50.01	41.50	43.72	51.88	49.89	41.51	43.47
LBL	51.80	49.69	41.54	43.76	51.93	49.86	41.27	43.33
LBL+LEN	51.90	49.76	41.41	43.70	52.42	50.29	40.93	43.00
LBL+LEN+CLM	52.61	50.51	40.77	43.03	52.60	50.56	40.69	42.81

Table 1: BLEU and TER percentage scores on MT06 and MT08 Arabic-to-English newswire sets.

other hand, one string-to-dependency rule may split into several dependency-to-dependency rules due to different source dependency structures. The size of the dependency-to-dependency rule set is slightly smaller than the size of the string-to-dependency rule set.

Tables 1 and 2 show the BLEU and TER percentage scores on MT06 and MT08 for Arabic-to-English and Chinese-to-English translation respectively. The context LM feature, the length feature and the syntax label feature all produce a small improvement for most of the conditions. When we combined the three features, we observed significant improvements over the baseline. For Arabic-to-English MT, the LBL+LEN+CLM system improved lower-cased BLEU by 2.0 on MT06 and 1.7 on MT08 on decoding output. For Chinese-to-English MT, the improvements in lower-cased BLEU were 1.0 on MT06 and 0.8 on MT08. After re-scoring, the improvements became smaller, but still noticeable, ranging from 0.7 to 1.4. TER scores were also improved noticeably for all conditions, suggesting there was no metric specific over-tuning.

Surprisingly, source dependency LM did not provide any improvement over the baseline. There are two possible reasons for this. One is that the source and target parse trees were generated by two stand-alone parsers, which may cause incompatible structures on the source and target sides. By applying the *well-formed* constraints on both sides, a lot of useful transfer rules are discarded. A bi-lingual parser, trained on parallel treebanks recently made available to the NLP community, may overcome this problem. The other is that the search space of dependency-to-dependency decoding is much larger, since we need to add source dependency information into the chart parsing states. We will explore techniques to address these problems in the future.

4 Discussion

Linguistic information has been widely used in SMT. For example, in (Wang et al., 2007), syntactic structures were employed to reorder the source language as a pre-processing step for phrase-based decoding. In (Koehn and Hoang, 2007), shallow syntactic analysis such as POS tagging and morphological analysis were incorporated in a phrasal decoder.

In ISI's syntax-based system (Galley et al., 2006) and CMU's Hiero extension (Venugopal et al., 2007), non-terminals in translation rules have labels, which must be respected by substitutions during decoding. In (Post and Gildea, 2008; Shen et al., 2008), target trees were employed to improve the scoring of translation theories. Marton and Resnik (2008) introduced features defined on constituent labels to improve the Hiero system (Chiang, 2005). However, due to the limitation of MER training, only part of the feature space could used in the system. This problem was fixed by

	MT06				MT08			
Model	**BLEU**		**TER**		**BLEU**		**TER**	
	lower	**mixed**	**lower**	**mixed**	**lower**	**mixed**	**lower**	**mixed**
Decoding (3-gram LM)								
BASE	37.44	35.62	54.64	56.47	33.05	31.26	56.79	58.69
SLM	37.30	35.48	54.24	55.90	33.03	31.00	56.59	58.46
CLM	37.66	35.81	53.45	55.19	32.97	31.01	55.99	57.77
LEN	38.09	36.26	53.98	55.81	33.23	31.34	56.51	58.41
LBL	38.37	36.53	54.14	55.99	33.25	31.34	56.60	58.49
LBL+LEN	38.36	36.59	53.95	55.60	33.72	31.83	56.79	58.65
LBL+LEN+CLM	38.41	36.57	53.83	55.70	33.83	31.79	56.55	58.51
Rescoring (5-gram LM)								
BASE	38.91	37.04	53.65	55.45	34.34	32.32	55.60	57.60
SLM	38.27	36.38	53.64	55.29	34.25	32.28	55.35	57.21
CLM	38.79	36.88	53.09	54.80	35.01	32.98	55.39	57.28
LEN	39.22	37.30	53.34	55.06	34.65	32.70	55.61	57.51
LBL	39.11	37.30	53.61	55.29	35.02	33.00	55.39	57.48
LBL+LEN	38.91	37.17	53.56	55.27	35.03	33.08	55.47	57.46
LBL+LEN+CLM	39.58	37.62	53.21	54.94	35.72	33.63	54.88	56.98

Table 2: BLEU and TER percentage scores on MT06 and MT08 Chinese-to-English newswire sets.

Chiang et al. (2008), which used an online learning method (Crammer and Singer, 2003) to handle a large set of features.

Most SMT systems assume that translation rules can be applied without paying attention to the sentence context. A few studies (Carpuat and Wu, 2007; Ittycheriah and Roukos, 2007; He et al., 2008; Hasan et al., 2008) addressed this defect by selecting the appropriate translation rules for an input span based on its context in the input sentence. The direct translation model in (Ittycheriah and Roukos, 2007) employed syntactic (POS tags) and context information (neighboring words) within a maximum entropy model to predict the correct transfer rules. A similar technique was applied by He et al. (2008) to improve the Hiero system.

Our model differs from previous work on the way in which linguistic and contextual information is used.

5 Conclusions and Future Work

In this paper, we proposed four new linguistic and contextual features for hierarchical decoding. The use of non-terminal labels, length distribution and context LM features gave rise to significant improvement on Arabic-to-English and Chinese-to-English translation on NIST MT06 and MT08 newswire data over a state-of-the-art string-to-dependency baseline. Unlike previous work, we employed robust probabilistic models to capture useful linguistic and contextual information. Our methods are more suitable for practical translation tasks.

In future, we will continue this work in two directions. We will employ a Gaussian model to unify various linguistic and contextual features. We will also improve the dependency-to-dependency method with a better bi-lingual parser.

Acknowledgments

This work was supported by DARPA/IPTO Contract No. HR0011-06-C-0022 under the GALE program.

References

R. P. Brent. 1973. *Algorithms for Minimization Without Derivatives.* Prentice-Hall.

I. Bulyko, S. Matsoukas, R. Schwartz, L. Nguyen, and J. Makhoul. 2007. Language model adaptation in machine translation from speech. In *Proceedings of the 32nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

M. Carpuat and D. Wu. 2007. Context-dependent phrasal translation lexicons for statistical machine translation. In *Proceedings of Machine Translation Summit XI*.

D. Chiang, Y. Marton, and P. Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference of Empirical Methods in Natural Language Processing*.

D. Chiang, K. Knight, and W. Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of the 2009 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, and I. Thayer. 2006. Scalable inference and training of context-rich syntactic models. In *COLING-ACL '06: Proceedings of 44th Annual Meeting of the Association for Computational Linguistics and 21st Int. Conf. on Computational Linguistics*.

J.-L. Gauvain and Chin-Hui Lee. 1994. Maximum a posteriori estimation for multivariate gaussian mixtureobservations of markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2).

S. Hasan, J. Ganitkevitch, H. Ney, and J. Andrés-Ferrer. 2008. Triplet lexicon models for statistical machine translation. In *Proceedings of the 2008 Conference of Empirical Methods in Natural Language Processing*.

Z. He, Q. Liu, and S. Lin. 2008. Improving statistical machine translation using lexicalized rule selection. In *Proceedings of COLING '08: The 22nd Int. Conf. on Computational Linguistics*.

A. Ittycheriah and S. Roukos. 2007. Direct translation model 2. In *Proceedings of the 2007 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

P. Koehn and H. Hoang. 2007. Factored translation models. In *Proceedings of the 2007 Conference of Empirical Methods in Natural Language Processing*.

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *COLING-ACL '06: Proceedings of 44th Annual Meeting of the Association for Computational Linguistics and 21st Int. Conf. on Computational Linguistics*.

Y. Marton and P. Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*.

M. Ostendorf, A. Kannan, S. Austin, O. Kimball, R. Schwartz, and J. R. Rohlicek. 1991. Integration of diverse recognition methodologies through reevaluation of nbest sentence hypotheses. In *Proceedings of the DARPA Workshop on Speech and Natural Language*.

K. Papineni, S. Roukos, and T. Ward. 2001. Bleu: a method for automatic evaluation of machine translation. IBM Research Report, RC22176.

M. Post and D. Gildea. 2008. Parsers as language models for statistical machine translation. In *The Eighth Conference of the Association for Machine Translation in the Americas*.

M. J. D. Powell. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2).

L. Shen, A. Sarkar, and F. J. Och. 2004. Discriminative reranking for machine translation. In *Proceedings of the 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

L. Shen, J. Xu, and R. Weischedel. 2008. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*.

C. Tillmann and T. Zhang. 2006. A discriminative global training algorithm for statistical mt. In *COLING-ACL '06: Proceedings of 44th Annual Meeting of the Association for Computational Linguistics and 21st Int. Conf. on Computational Linguistics*.

A. Venugopal, A. Zollmann, and S. Vogel. 2007. An efficient two-pass approach to synchronous-cfg driven statistical mt. In *Proceedings of the 2007 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

C. Wang, M. Collins, and P. Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proceedings of the 2007 Conference of Empirical Methods in Natural Language Processing*.

Active Learning by Labeling Features

Gregory Druck
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
gdruck@cs.umass.edu

Burr Settles
Dept. of Biostatistics & Medical Informatics
Dept. of Computer Sciences
University of Wisconsin
Madison, WI 53706
bsettles@cs.wisc.edu

Andrew McCallum
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
mccallum@cs.umass.edu

Abstract

Methods that learn from prior information about input features such as *generalized expectation (GE)* have been used to train accurate models with very little effort. In this paper, we propose an *active learning* approach in which the machine solicits "labels" on *features* rather than instances. In both simulated and real user experiments on two sequence labeling tasks we show that our active learning method outperforms passive learning with features as well as traditional active learning with instances. Preliminary experiments suggest that novel interfaces which intelligently solicit labels on multiple features facilitate more efficient annotation.

1 Introduction

The application of machine learning to new problems is slowed by the need for labeled training data. When output variables are structured, annotation can be particularly difficult and time-consuming. For example, when training a conditional random field (Lafferty et al., 2001) to extract fields such as rent, contact, features, and utilities from apartment classifieds, labeling 22 instances (2,540 tokens) provides only 66.1% accuracy.[1]

Recent work has used unlabeled data and limited prior information about input features to bootstrap accurate structured output models. For example, both Haghighi and Klein (2006) and Mann and McCallum (2008) have demonstrated results better than 66.1% on the *apartments* task described above using only a list of 33 highly discriminative features and the labels they indicate. However, these methods have only been applied in scenarios in which the user supplies such prior knowledge before learning begins.

[1] Averaged over 10 randomly selected sets of 22 instances.

In traditional *active learning* (Settles, 2009), the machine queries the user for only the labels of instances that would be most helpful to the machine. This paper proposes an active learning approach in which the user provides "labels" for input features, rather than instances. A labeled input feature denotes that a particular input feature, for example the word *call*, is highly indicative of a particular label, such as contact. Table 1 provides an excerpt of a feature active learning session.

In this paper, we advocate using generalized expectation (GE) criteria (Mann and McCallum, 2008) for learning with labeled features. We provide an alternate treatment of the GE objective function used by Mann and McCallum (2008) and a novel speedup to the gradient computation. We then provide a pool-based feature active learning algorithm that includes an option to skip queries, for cases in which a feature has no clear label. We propose and evaluate feature query selection algorithms that aim to reduce model uncertainty, and compare to several baselines. We evaluate our method using both real and simulated user experiments on two sequence labeling tasks. Compared to previous approaches (Raghavan and Allan, 2007), our method can be used for both classification and structured tasks, and the feature query selection methods we propose perform better.

We use experiments with simulated labelers on real data to extensively compare feature query selection algorithms and evaluate on multiple random splits. To make these simulations more realistic, the effort required to perform different labeling actions is estimated from additional experiments with real users. The results show that active learning with features outperforms both passive learning with features and traditional active learning with instances.

In the user experiments, each annotator actively labels instances, actively labels features one at a time, and actively labels batches of features orga-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 81–90,
Singapore, 6-7 August 2009.

accuracy 46.5 → 60.5	
feature	label
*PHONE**	contact
call	contact
deposit	rent
month	rent
pets	restrict.
lease	rent
appointment	contact
parking	features
*EMAIL**	contact
information	contact

accuracy 60.5 → 67.1	
feature	label
water	utilities
close	neighbor.
garbage	utilities
included	utilities
	features
shopping	neighbor.
bart	neighbor.
downtown	neighbor.
*TIME**	contact
bath	size

Table 1: Two iterations of feature active learning. Each table shows the features labeled, and the resulting change in accuracy. Note that the word *included* was labeled as both utilities and features, and that $*$ denotes a regular expression feature.

nized using a "grid" interface. The results support the findings of the simulated experiments and provide evidence that the "grid" interface can facilitate more efficient annotation.

2 Conditional Random Fields

In this section we describe the underlying probabilistic model for all methods in this paper. We focus on sequence labeling, though the described methods could be applied to other structured output or classification tasks. We model the probability of the label sequence $\mathbf{y} \in \mathcal{Y}^n$ conditioned on the input sequence $\mathbf{x} \in \mathcal{X}^n$, $p(\mathbf{y}|\mathbf{x};\theta)$ using first-order linear-chain conditional random fields (CRFs) (Lafferty et al., 2001). This probability is

$$p(\mathbf{y}|\mathbf{x};\theta) = \frac{1}{Z_\mathbf{x}} \exp\big(\sum_i \sum_j \theta_j f_j(y_i, y_{i+1}, \mathbf{x}, i)\big),$$

where $Z_\mathbf{x}$ is the partition function and feature functions f_j consider the entire input sequence and at most two consecutive output variables. The most probable output sequence and transition marginal distributions can be computed using variants of Viterbi and forward-backward.

Provided a training data distribution $\tilde{p}$, we estimate CRF parameters by maximizing the conditional log likelihood of the training data.

$$\mathcal{L}(\theta) = E_{\tilde{p}(\mathbf{x},\mathbf{y})}[\log p(\mathbf{y}|\mathbf{x};\theta)]$$

We use numerical optimization to maximize $\mathcal{L}(\theta)$, which requires the gradient of $\mathcal{L}(\theta)$ with respect to the parameters. It can be shown that the partial derivative with respect to parameter j is equal to the difference between the empirical expectation of F_j and the model expectation of F_j, where $F_j(\mathbf{y},\mathbf{x}) = \sum_i f_j(y_i, y_{i+1}, \mathbf{x}, i)$.

$$\frac{\partial}{\partial\theta_j}\mathcal{L}(\theta) = E_{\tilde{p}(\mathbf{x},\mathbf{y})}[F_j(\mathbf{y},\mathbf{x})] - E_{\tilde{p}(\mathbf{x})}[E_{p(\mathbf{y}|\mathbf{x};\theta)}[F_j(\mathbf{y},\mathbf{x})]].$$

We also include a zero-mean variance $\sigma^2 = 10$ Gaussian prior on parameters in all experiments.[2]

2.1 Learning with missing labels

The training set may contain partially labeled sequences. Let $\mathbf{z}$ denote missing labels. We estimate parameters with this data by maximizing the marginal log-likelihood of the observed labels.

$$\mathcal{L}_{MML}(\theta) = E_{\tilde{p}(\mathbf{x},\mathbf{y})}[\log \sum_\mathbf{z} p(\mathbf{y},\mathbf{z}|\mathbf{x};\theta)]$$

We refer to this training method as *maximum marginal likelihood* (MML); it has also been explored by Quattoni et al. (2007).

The gradient of $\mathcal{L}_{MML}(\theta)$ can also be written as the difference of two expectations. The first is an expectation over the empirical distribution of $\mathbf{x}$ and $\mathbf{y}$, and the model distribution of $\mathbf{z}$. The second is a double expectation over the empirical distribution of $\mathbf{x}$ and the model distribution of $\mathbf{y}$ and $\mathbf{z}$.

$$\frac{\partial}{\partial\theta_j}\mathcal{L}_{MML}(\theta) = E_{\tilde{p}(\mathbf{x},\mathbf{y})}[E_{p(\mathbf{z}|\mathbf{y},\mathbf{x};\theta)}[F_j(\mathbf{y},\mathbf{z},\mathbf{x})]] - E_{\tilde{p}(\mathbf{x})}[E_{p(\mathbf{y},\mathbf{z}|\mathbf{x};\theta)}[F_j(\mathbf{y},\mathbf{z},\mathbf{x})]].$$

We train models using $\mathcal{L}_{MML}(\theta)$ with expected gradient (Salakhutdinov et al., 2003).

To additionally leverage unlabeled data, we compare with *entropy regularization* (ER). ER adds a term to the objective function that encourages confident predictions on unlabeled data. Training of linear-chain CRFs with ER is described by Jiao et al. (2006).

3 Generalized Expectation Criteria

In this section, we give a brief overview of *generalized expectation criteria* (GE) (Mann and McCallum, 2008; Druck et al., 2008) and explain how we can use GE to learn CRF parameters with estimates of feature expectations and unlabeled data.

GE criteria are terms in a parameter estimation objective function that express preferences on the

[2] 10 is a default value that works well in many settings.

value of a model expectation of some function. Given a score function S, an empirical distribution $\tilde{p}(\mathbf{x})$, a model distribution $p(\mathbf{y}|\mathbf{x};\theta)$, and a constraint function $G_k(\mathbf{x},\mathbf{y})$, the value of a GE criterion is $\mathcal{G}(\theta) = S(E_{\tilde{p}(\mathbf{x})}[E_{p(\mathbf{y}|\mathbf{x};\theta)}[G_k(\mathbf{x},\mathbf{y})]])$.

GE provides a flexible framework for parameter estimation because each of these elements can take an arbitrary form. The most important difference between GE and other parameter estimation methods is that it does not require a one-to-one correspondence between constraint functions G_k and model feature functions. We leverage this flexibility to estimate parameters of feature-rich CRFs with a very small set of expectation constraints.

Constraint functions G_k can be normalized so that the sum of the expectations of a set of functions is 1. In this case, S may measure the divergence between the expectation of the constraint function and a target expectation $\hat{G}_k$.

$$\mathcal{G}(\theta) = \hat{G}_k \log(E[G_k(\mathbf{x},\mathbf{y})]), \tag{1}$$

where $E[G_k(\mathbf{x},\mathbf{y})] = E_{\tilde{p}(\mathbf{x})}[E_{p(\mathbf{y}|\mathbf{x};\theta)}[G_k(\mathbf{x},\mathbf{y})]]$.

It can be shown that the partial derivative of $\mathcal{G}(\theta)$ with respect to parameter j is proportional to the predicted covariance between the model feature function F_j and the constraint function G_k.[3]

$$\frac{\partial}{\partial\theta_j}\mathcal{G}(\theta) = \frac{\hat{G}_k}{E[G_k(\mathbf{x},\mathbf{y})]}\times \tag{2}$$
$$\Big(E_{\tilde{p}(\mathbf{x})}\big[E_{p(\mathbf{y}|\mathbf{x};\theta)}[F_j(\mathbf{x},\mathbf{y})G_k(\mathbf{x},\mathbf{y})]$$
$$- E_{p(\mathbf{y}|\mathbf{x};\theta)}[F_j(\mathbf{x},\mathbf{y})]E_{p(\mathbf{y}|\mathbf{x};\theta)}[G_k(\mathbf{x},\mathbf{y})]\big]\Big)$$

The partial derivative shows that GE learns parameter values for model feature functions based on their predicted covariance with the constraint functions. GE can thus be interpreted as a bootstrapping method that uses the limited training signal to learn about parameters for related model feature functions.

3.1 Learning with feature-label distributions

Mann and McCallum (2008) apply GE to a linear-chain, first-order CRF. In this section we provide an alternate treatment that arrives at the same objective function from the general form described in the previous section.

Often, feature functions in a first-order linear-chain CRF f are binary, and are the conjunction of an observational test $q(\mathbf{x},i)$ and a label pair test $\mathbf{1}_{\{y_i=y',y_{i+1}=y''\}}$.[4]

$$f(y_i,y_{i+1},\mathbf{x},i) = \mathbf{1}_{\{y_i=y',y_{i+1}=y''\}}q(\mathbf{x},i)$$

The constraint functions G_k we use here decompose and operate similarly, except that they only include a test for a single label. Single label constraints are easier for users to estimate and make GE training more efficient. Label transition structure can be learned automatically from single label constraints through the covariance-based parameter update of Equation 2. For convenience, we can write G_{yk} to denote the constraint function that combines observation test k with a test for label y. We also add a normalization constant $C_k = E_{\tilde{p}(\mathbf{x})}[\sum_i q_k(\mathbf{x},i)]$,

$$G_{yk}(\mathbf{x},\mathbf{y}) = \sum_i \frac{1}{C_k}\mathbf{1}_{\{y_i=y\}}q_k(\mathbf{x},i)$$

Under this construction the expectation of G_{yk} is the predicted conditional probability that the label at some arbitrary position i is y when the observational test at i succeeds, $\tilde{p}(y_i{=}y|q_k(\mathbf{x},i){=}1;\theta)$.

If we have a set of constraint functions $\{G_{yk} : y \in \mathcal{Y}\}$, and we use the score function in Equation 1, then the GE objective function specifies the minimization of the KL divergence between the model and target distributions over labels conditioned on the success of the observational test. In general the objective function will consist of many such KL divergence penalties.

Computing the first term of the covariance in Equation 2 requires a marginal distribution over three labels, two of which will be consecutive, but the other of which could appear anywhere in the sequence. We can compute this marginal using the algorithm of Mann and McCallum (2008). As previously described, this algorithm is $O(n|\mathcal{Y}|^3)$ for a sequence of length n. However, we make the following novel observation: we do not need to compute the extra lattices for feature label pairs with $\hat{G}_{yk} = 0$, since this makes Equation 2 equal to zero. In Mann and McCallum (2008), probabilities were smoothed so that $\forall_y\ \hat{G}_{yk} > 0$. If we assume that only a small number of labels m have non-zero probability, then the time complexity of the gradient computation is $O(nm|\mathcal{Y}|^2)$. In this paper typically $1 \leq m \leq 4$, while $|\mathcal{Y}|$ is 11 or 13.

[3] If we use squared error for S, the partial derivative is the covariance multiplied by $2(\hat{G}_k - E[G_k(\mathbf{x},\mathbf{y})])$.

[4] We this notation for an indicator function that returns 1 if the condition in braces is satisfied, and 0 otherwise.

In experiments in this paper, using this optimization does not significantly affect final accuracy.

We use numerical optimization to estimate model parameters. In general GE objective functions are not convex. Consequently, we initialize 0th-order CRF parameters using a sliding window logistic regression model trained with GE. We also include a Gaussian prior on parameters with $\sigma^2 = 10$ in the objective function.

3.2 Learning with labeled features

The training procedure described above requires a set of observational tests or input features with target distributions over labels. Estimating a distribution could be a difficult task for an annotator. Consequently, we abstract away from specifying a distribution by allowing the user to assign labels to features (c.f. Haghighi and Klein (2006) , Druck et al. (2008)). For example, we say that the word feature *call* has label contact. A label for a feature simply indicates that the feature is a good indicator of the label. Note that features can have multiple labels, as does *included* in the active learning session shown in Table 1. We convert an input feature with a set of labels L into a distribution by assigning probability $1/|L|$ for each $l \in L$ and probability 0 for each $l \notin L$. By assigning 0 probability to labels $l \notin L$, we can use the speed-up described in the previous section.

3.3 Related Work

Other proposed learning methods use labeled features to label unlabeled data. The resulting partially-labeled corpus can be used to train a CRF by maximizing MML. Similarly, *prototype-driven learning* (PDL) (Haghighi and Klein, 2006) optimizes the joint marginal likelihood of data labeled with *prototype* input features for each label. Additional features that indicate similarity to the prototypes help the model to generalize. In a previous comparison between GE and PDL (Mann and McCallum, 2008), GE outperformed PDL without the extra similarity features, whose construction may be problem-specific. GE also performed better when supplied accurate label distributions.

Additionally, both MML and PDL do not naturally generalize to learning with features that have multiple labels or distributions over labels, as in these scenarios labeling the unlabeled data is not straightforward. In this paper, we attempt to address this problem using a simple heuristic: when there are multiple choices for a token's label, sample a label. In Section 5 we use this heuristic with MML, but in general obtain poor results.

Raghavan and Allan (2007) also propose several methods for learning with labeled features, but in a previous comparison GE gave better results (Druck et al., 2008). Additionally, the generalization of these methods to structured output spaces is not straightforward. Chang et al. (2007) present an algorithm for learning with constraints, but this method requires users to set weights by hand. We plan to explore the use of the recently developed related methods of Bellare et al. (2009), Graça et al. (2008), and Liang et al. (2009) in future work. Druck et al. (2008) provide a survey of other related methods for learning with labeled input features.

4 Active Learning by Labeling Features

Feature active learning, presented in Algorithm 1, is a *pool-based* active learning algorithm (Lewis and Gale, 1994) (with a pool of features rather than instances). The novel components of the algorithm are an option to skip a query and the notion that skipping and labeling have different costs. The option to skip is important when using feature queries because a user may not know how to label some features. In each iteration the model is retrained using the train procedure, which takes as input a set of labeled features $\mathcal{C}$ and unlabeled data distribution $\tilde{p}$. For the reasons described in Section 3.3, we advocate using GE for the train procedure. Then, while the iteration cost c is less than the maximum cost c_{max}, the feature query q that maximizes the query selection metric ϕ is selected. The accept function determines whether the labeler will label q. If q is labeled, it is added to the set of labeled features $\mathcal{C}$, and the label cost c_{label} is added to c. Otherwise, the skip cost c_{skip} is added to c. This process continues for N iterations.

4.1 Feature query selection methods

In this section we propose feature query selection methods ϕ. Queries with a higher scores are considered better candidates. Note again that by features we mean observational tests $q_k(\mathbf{x}, i)$. It is also important to note these are not *feature selection* methods since we are determining the features for which supervisory feedback will be most helpful to the model, rather than determining which features will be part of the model.

Algorithm 1 Feature Active Learning

Input: empirical distribution $\tilde{p}$, initial feature constraints $\mathcal{C}$, label cost c_{label}, skip cost c_{skip}, max cost per iteration c_{max}, max iterations N
Output: model parameters θ
for $i = 1$ **to** N **do**
  $\theta = \text{train}(\tilde{p}, \mathcal{C})$
  $c = 0$
  while $c < c_{max}$ **do**
    $q = \text{argmax}_{q_k}\, \phi(q_k)$
    if $\text{accept}(q)$ **then**
      $\mathcal{C} = \mathcal{C} \cup \text{label}(q)$
      $c = c + c_{label}$
    else
      $c = c + c_{skip}$
    end if
  end while
end for
$\theta = \text{train}(\tilde{p}, \mathcal{C})$

We propose to select queries that provide the largest reduction in model uncertainty. We notate possible responses to a query q_k as $\hat{g}$. The **Expected Information Gain (EIG)** of a query is the expectation of the reduction in model uncertainty over all possible responses. Mathematically, IG is

$$\phi_{EIG}(q_k) = E_{p(\hat{g}|q_k;\theta)}[E_{\tilde{p}(\mathbf{x})}[H(p(\mathbf{y}|\mathbf{x};\theta) - H(p(\mathbf{y}|\mathbf{x};\theta_{\hat{g}})]],$$

where $\theta_{\hat{g}}$ are the new model parameters if the response to q_k is $\hat{g}$. Unfortunately, this method is computationally intractable. Re-estimating $\theta_{\hat{g}}$ will typically involve retraining the model, and doing this for each possible query-response pair is prohibitively expensive for structured output models. Computing the expectation over possible responses is also difficult, as in this paper users may provide a set of labels for a query, and more generally $\hat{g}$ could be a distribution over labels.

Instead, we propose a tractable strategy for reducing model uncertainty, motivated by traditional uncertainty sampling (Lewis and Gale, 1994). We assume that when a user responds to a query, the reduction in uncertainty will be equal to the **Total Uncertainty (TU)**, the sum of the marginal entropies at the positions where the feature occurs.

$$\phi_{TU}(q_k) = \sum_i \sum_j q_k(\mathbf{x}_i, j) H(p(y_j|\mathbf{x}_i;\theta))$$

Total uncertainty, however, is highly biased towards selecting frequent features. A mean uncertainty variant, normalized by the feature's count, would tend to choose very infrequent features. Consequently we propose a tradeoff between the two extremes, called **weighted uncertainty (WU)**, that scales the mean uncertainty by the log count of the feature in the corpus.

$$\phi_{WU}(q_k) = \log(C_k)\frac{\phi_{TU}(q_k)}{C_k}.$$

Finally, we also suggest an uncertainty-based metric called **diverse uncertainty (DU)** that encourages diversity among queries by multiplying TU by the mean dissimilarity between the feature and previously labeled features. For sequence labeling tasks, we can measure the relatedness of features using distributional similarity.[5]

$$\phi_{DU}(q_k) = \phi_{TU}(q_k)\frac{1}{|\mathcal{C}|}\sum_{j\in\mathcal{C}} 1 - \text{sim}(q_k, q_j)$$

We contrast the notion of uncertainty described above with another type of uncertainty: the entropy of the predicted label distribution for the feature, or **expectation uncertainty (EU)**. As above we also multiply by the log feature count.

$$\phi_{EU}(q_k) = \log(C_k) H(\tilde{p}(y_i = y | q_k(\mathbf{x}, i) = 1; \theta))$$

EU is flawed because it will have a large value for non-discriminative features.

The methods described above require the model to be retrained between iterations. To verify that this is necessary, we compare against query selection methods that only consider the previously labeled features. First, we consider a feature query selection method called **coverage (cov)** that aims to select features that are dissimilar from existing labeled features, increasing the labeled features' "coverage" of the feature space. In order to compensate for choosing very infrequent features, we multiply by the log count of the feature.

$$\phi_{cov}(q_k) = \log(C_k)\frac{1}{|\mathcal{C}|}\sum_{j\in\mathcal{C}} 1 - \text{sim}(q_k, q_j)$$

Motivated by the feature query selection method of Tandem Learning (Raghavan and Allan, 2007) (see Section 4.2 for further discussion), we consider a feature selection metric **similarity (sim)** that is the maximum similarity to a labeled feature, weighted by the log count of the feature.

$$\phi_{sim}(q_k) = \log(C_k) \max_{j\in\mathcal{C}} \text{sim}(q_k, q_j)$$

[5] $\text{sim}(q_k, q_j)$ returns the cosine similarity between context vectors of words occurring in a window of ± 3.

Features similar to those already labeled are likely to be discriminative, and therefore likely to be labeled (rather than skipped). However, insufficient diversity may also result in an inaccurate model, suggesting that *coverage* should select more useful queries than *similarity*.

Finally, we compare with several passive baselines. **Random (rand)** assigns scores to features randomly. **Frequency (freq)** scores input features using their frequency in the training data.

$$\phi_{freq}(q_k) = \sum_i \sum_j q_k(\mathbf{x}_i, j)$$

Top LDA (LDA) selects the top words from 50 topics learned from the unlabeled data using latent Dirichlet allocation (LDA) (Blei et al., 2003). More specifically, the words w generated by each topic t are ranked using the conditional probability $p(w|t)$. The word feature is assigned its maximum rank across all topics.

$$\phi_{LDA}(q_k) = \max_t \ \mathrm{rank}_{\mathrm{LDA}}(q_k, t)$$

This method will select useful features if the topics discovered are relevant to the task. A similar heuristic was used by Druck et al. (2008).

4.2 Related Work

Tandem Learning (Raghavan and Allan, 2007) is an algorithm that combines feature and instance active learning for classification. The algorithm iteratively queries the user first for instance labels, then for feature labels. Feature queries are selected according to their co-occurrence with important model features and previously labeled features. As noted in Section 3.3, GE is preferable to the methods Tandem Learning uses to learn with labeled features. We address the mixing of feature and instance queries in Section 4.3.

In order to better understand differences in feature query selection methodology, we proposed a feature query selection method motivated[6] by the method used in Tandem Learning in Section 4.1. However, this method performs poorly in the experiments in Section 5.

Liang et al. (2009) simultaneously developed a method for learning with and actively selecting *measurements*, or target expectations with associated noise. The measurement selection method proposed by Liang et al. (2009) is based on Bayesian experimental design and is similar to the expected information gain method described above. Consequently this method is likely to be intractable for real applications. Note that Liang et al. (2009) only use this method in synthetic experiments, and instead use a method similar to total uncertainty for experiments in part-of-speech tagging. Unlike the experiments presented in this paper, Liang et al. (2009) conduct only simulated active learning experiments and do not consider skipping queries.

Sindhwani (Sindhwani et al., 2009) simultaneously developed an active learning method that queries for both instance and feature labels that are then used in a graph-based learning algorithm. They find that querying certain features outperforms querying uncertain features, but this is likely because their query selection method is similar to the expectation uncertainty method described above, and consequently non-discriminative features may be queried often (see also the discussion in Section 4.1). It is also not clear how this graph-based training method would generalize to structured output spaces.

4.3 Expectation Constraint Active Learning

Throughout this paper, we have focussed on labeling input features. However, the proposed methods generalize to queries for expectation estimates of arbitrary functions, for example queries for the label distributions for input features, labels for instances (using a function that is non-zero only for a particular instance), partial labels for instances, and class priors. The uncertainty-based query selection methods described in Section 4.1 apply naturally to these new query types. Importantly this framework would allow principled mixing of different query types, instead of alternating between them as in Tandem Learning (Raghavan and Allan, 2007). When mixing queries, it will be important to use different costs for different annotation types (Vijayanarasimhan and Grauman, 2008), and estimate the probability of obtaining a useful response to a query. We plan to pursue these directions in future work. This idea was also proposed by Liang et al. (2009), but no experiments with mixed active learning were presented.

[6]The query selection method of Raghavan and Allan (2007) requires a stack that is modified between queries within each iteration. Here query scores are only updated after each iteration of labeling.

5 Simulated User Experiments

In this section we experiment with an automated oracle labeler. When presented an instance query, the oracle simply provides the true labels. When presented a feature query, the oracle first decides whether to skip the query. We have found that users are more likely to label features that are relevant for only a few labels. Therefore, the oracle labels a feature if the entropy of its per occurrence label expectation, $\mathrm{H}(\tilde{p}(y_i = y|q_k(\mathbf{x}, i) = 1; \theta)) \leq 0.7$. The oracle then labels the feature using a heuristic: label the feature with the label whose expectation is highest, as well as any label whose expectation is at least half as large.

We estimate the effort of different labeling actions with preliminary experiments in which we observe users labeling data for ten minutes. Users took an average of 4 seconds to label a feature, 2 seconds to skip a feature, and 0.7 seconds to label a token. We setup experiments such that each iteration simulates one minute of labeling by setting $c_{max} = 60$, $c_{skip} = 2$ and $c_{label} = 4$. For instance active learning, we use Algorithm 1 but without the skip option, and set $c_{label} = 0.7$. We use $N = 10$ iterations, so the entire experiment simulates 10 minutes of annotation time. For efficiency, we consider the 500 most frequent unlabeled features in each iteration. To start, ten randomly selected seed labeled features are provided.

We use **random (rand)** selection, **uncertainty sampling (US)** (using sequence entropy, normalized by sequence length) and **information density (ID)** (Settles and Craven, 2008) to select instance queries. We use Entropy Regularization (ER) (Jiao et al., 2006) to leverage unlabeled instances.[7] We weight the ER term by choosing the best[8] weight in $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$ multiplied by $\frac{\#labeled}{\#unlabeled}$ for each data set and query selection method. Seed instances are provided such that the simulated labeling time is equivalent to labeling 10 features.

We evaluate on two sequence labeling tasks. The *apartments* task involves segmenting 300 apartment classified ads into 11 fields including features, rent, neighborhood, and contact. We use the same feature processing as Haghighi and Klein (2006), with the addition of context features in a window of ± 3. The *cora references* task is to extract 13 BibTeX fields such as author and booktitle from 500 research paper references. We use a standard set of word, regular expressions, and lexicon features, as well as context features in a window of ± 3. All results are averaged over ten random 80:20 splits of the data.

[7] Results using self-training instead of ER are similar.

[8] As measured by test accuracy, giving ER an advantage.

method	apartments		cora	
	mean	**final**	**mean**	**final**
ER rand	48.1	53.6	75.9	81.1
ER US	51.7	57.9	76.0	83.2
ER ID	51.4	56.9	75.9	83.1
MML rand	47.7	51.2	58.6	64.6
MML WU	57.6	60.8	61.0	66.2
GE rand	59.0	64.8*	77.6	83.7
GE freq	66.5*	71.6*	68.6	79.8
GE LDA	65.7*	71.4*	74.9	85.0
GE cov	68.2*†	72.6*	73.5	83.3
GE sim	57.8	65.9*	67.1	79.2
GE EU	66.5*	71.6*	68.6	79.8
GE TU	70.1*†	**73.6***†	76.9	**88.2***†
GE WU	**71.6***†	**74.6***†	**80.3***†	**88.1***†
GE DU	70.5*†	**74.4***†	78.4*	**87.5***†

Table 2: Mean and final token accuracy results. A * or † denotes that a GE method significantly outperforms all non-GE or passive GE methods, respectively. Bold entries significantly outperform all others. Methods in *italics* are passive.

5.1 Results

Table 2 presents mean (across all iterations) and final token accuracy results. On the *apartments* task, GE methods greatly outperform MML[9] and ER methods. Each uncertainty-based GE method also outperforms all passive GE methods. On the *cora* task, only GE with *weighted uncertainty* significantly outperforms ER and passive GE methods in terms of *mean* accuracy, but all uncertainty-based GE methods provide higher final accuracy. This suggests that on the *cora* task, active GE methods are performing better in later iterations. Figure 1, which compares the learning curves of the best performing methods of each type, shows this phenomenon. Further analysis reveals that the uncertainty-based methods are choosing frequent features that are more likely to be skipped than those selected randomly in early iterations.

We next compare with the results of related methods published elsewhere. We cannot make claims about statistical significance, but the results

[9] Only the best MML results are shown.

illustrate the competitiveness of our method. The 74.6% final accuracy on *apartments* is higher than any result obtained by Haghighi and Klein (2006) (the highest is 74.1%), higher than the *supervised* HMM results reported by Grenager et al. (2005) (74.4%), and matches the results of Mann and McCallum (2008) with GE with more accurate sampled label distributions and 10 labeled examples. Chang et al. (2007) only obtain better results than 88.2% on *cora* when using 300 labeled examples (two hours of estimated annotation time), 5000 additional unlabeled examples, and extra test time inference constraints. Note that obtaining these results required only 10 simulated minutes of annotation time, and that GE methods are provided no information about the label transition matrix.

6 User Experiments

Another advantage of feature queries is that feature names are concise enough to be browsed, rather than considered individually. This allows the design of improved interfaces that can further increase the speed of feature active learning. We built a prototype interface that allows the user to quickly browse many candidate features. The features are split into groups of five features each. Each group contains features that are related, as measured by distributional similarity. The features within each group are sorted according to the active learning metric. This interface, displayed in Figure 3, may be useful because features in the same group are likely to have the same label.

We conduct three types of experiments. First, a user labels instances selected by *information density*, and models are trained using ER. The instance labeling interface allows the user to label tokens quickly by extending the current selection one token at a time and only requiring a single keystroke to label an entire segment. Second, the user labels features presented one-at-a-time by *weighted uncertainty*, and models are trained using GE. To aid the user in understanding the function of the feature quickly, we provide several examples of the feature occurring in context and the model's current predicted label distribution for the feature. Finally, the user labels features organized using the grid interface described in the previous paragraph. *Weighted uncertainty* is used to sort feature queries within each group, and GE is used to train models. Each iteration of labeling lasts two minutes, and there are five iterations. Retraining with ER between iterations takes an average of 5 minutes on *cora* and 3 minutes on *apartments*. With GE, the retraining times are on average 6 minutes on *cora* and 4 minutes on *apartments*. Consequently, even when viewed with *total time*, rather than *annotation time*, feature active learning is beneficial. While waiting for models to retrain, users can perform other tasks.

Figure 2 displays the results. User 1 labeled *apartments* data, while Users 2 and 3 labeled *cora* data. User 1 was able to obtain much better results with feature labeling than with instance labeling, but performed slightly worse with the grid interface than with the serial interface. User 1 commented that they found the label definitions for *apartments* to be imprecise, so the other experiments were conducted on the *cora* data. User 2 obtained better results with feature labeling than instance labeling, and obtained higher mean accuracy with the grid interface. User 3 was much better at labeling features than instances, and performed especially well using the grid interface.

7 Conclusion

We proposed an active learning approach in which features, rather than instances, are labeled. We presented an algorithm for active learning with features and several feature query selection methods that approximate the expected reduction in model uncertainty of a feature query. In simulated experiments, active learning with features outperformed passive learning with features, and uncertainty-based feature query selection outperformed other baseline methods. In both simulated and real user experiments, active learning with features outperformed passive and active learning with instances. Finally, we proposed a new labeling interface that leverages the conciseness of feature queries. User experiments suggested that this grid interface can improve labeling efficiency.

Acknowledgments

We thank Kedar Bellare for helpful discussions and Gaurav Chandalia for providing code. This work was supported in part by the Center for Intelligent Information Retrieval and the Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249. The second author was supported by a grant from National Human Genome Research Institute. Any opinions, findings and conclusions or recommendations are the authors' and do not necessarily reflect those of the sponsor.

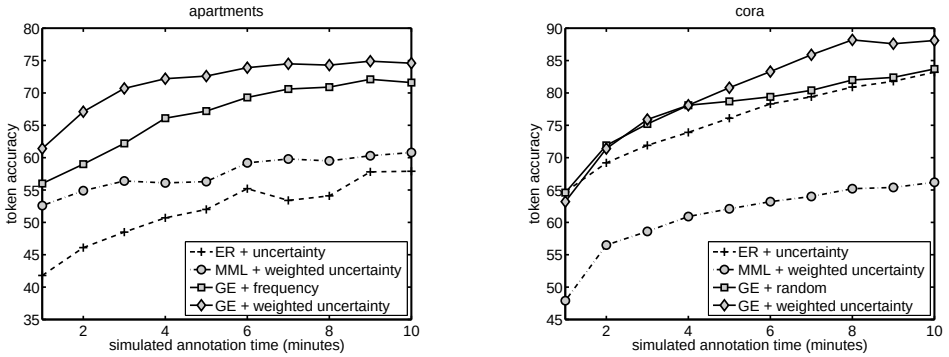


Figure 1: Token accuracy vs. time for best performing ER, MML, passive GE, and active GE methods.

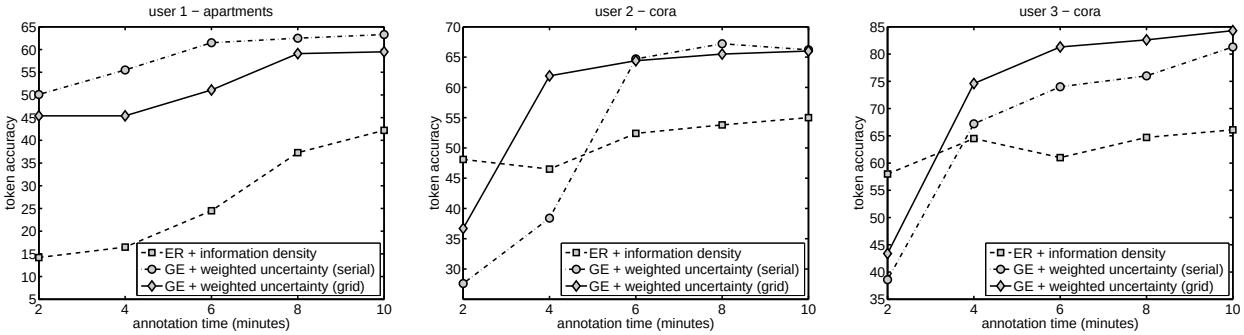


Figure 2: User experiments with instance labeling and feature labeling with the *serial* and *grid* interfaces.

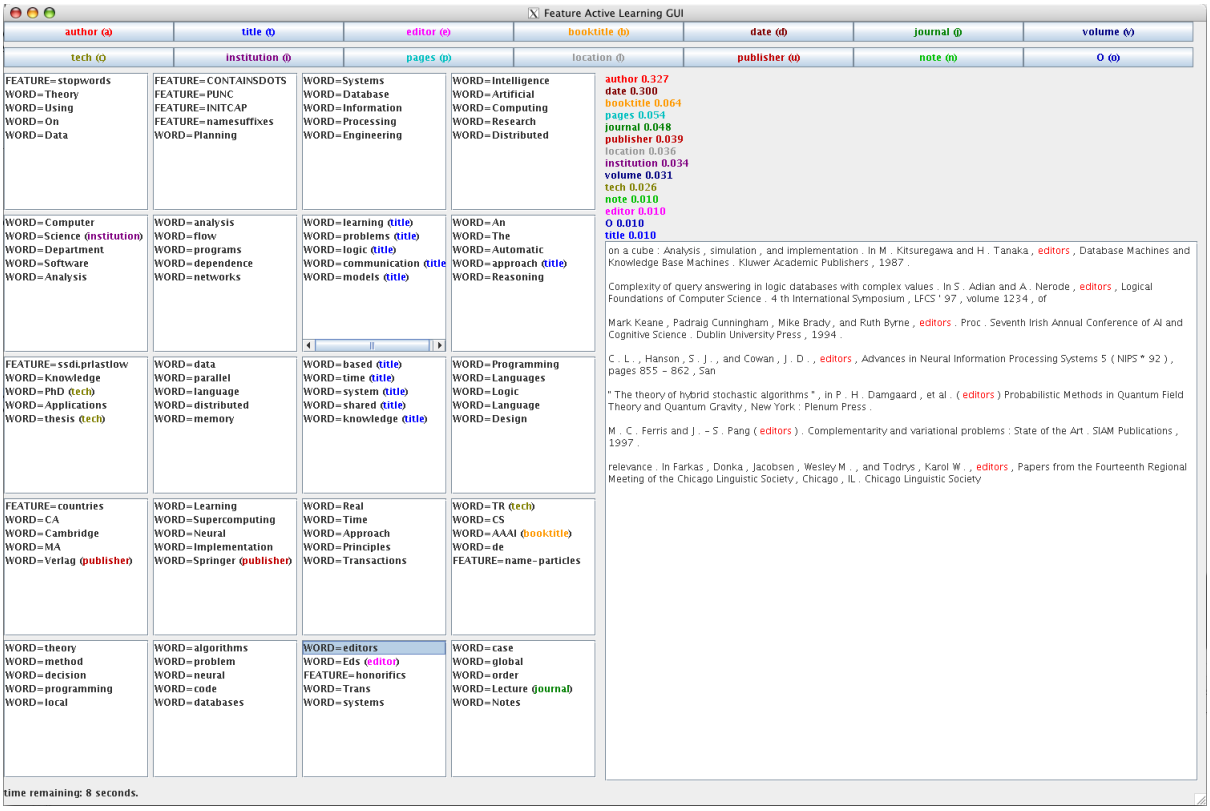


Figure 3: Grid feature labeling interface. Boxes on the left contain groups of features that appear in similar contexts. Features in the same group often receive the same label. On the right, the model's current expectation and occurrences of the selected feature in context are displayed.

References

Kedar Bellare, Gregory Druck, and Andrew McCallum. 2009. Alternating projections for learning with expectation constraints. In *UAI*.

David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *ACL*, pages 280–287.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *SIGIR*.

Joao Graça, Kuzman Ganchev, and Ben Taskar. 2008. Expectation maximization and posterior constraints. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press.

Trond Grenager, Dan Klein, and Christopher D. Manning. 2005. Unsupervised learning of field segmentation models for information extraction. In *ACL*.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *HTL-NAACL*.

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *ACL*, pages 209–216.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR*, pages 3–12, New York, NY, USA. Springer-Verlag New York, Inc.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning from measurements in exponential families. In *ICML*.

Gideon Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *ACL*.

A. Quattoni, S. Wang, L.-P Morency, M. Collins, and T. Darrell. 2007. Hidden conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1848–1852, October.

Hema Raghavan and James Allan. 2007. An interactive algorithm for asking and incorporating feature feedback into support vector machines. In *SIGIR*, pages 79–86.

Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. 2003. Optimization with em and expectation-conjugate-gradient. In *ICML*, pages 672–679.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP*.

Burr Settles. 2009. Active learning literature survey. Technical Report 1648, University of Wisconsin - Madison.

Vikas Sindhwani, Prem Melville, and Richard D. Lawrence. 2009. Uncertainty sampling and transductive experimental design for active dual supervision. In *ICML*.

Sudheendra Vijayanarasimhan and Kristen Grauman. 2008. Multi-level active prediction of useful image annotations for recognition. In *NIPS*.

Efficient kernels for sentence pair classification

Fabio Massimo Zanzotto
DISP
University of Rome "Tor Vergata"
Via del Politecnico 1
00133 Roma, Italy
zanzotto@info.uniroma2.it

Lorenzo Dell'Arciprete
University of Rome "Tor Vergata"
Via del Politecnico 1
00133 Roma, Italy
lorenzo.dellarciprete@gmail.com

Abstract

In this paper, we propose a novel class of graphs, the tripartite directed acyclic graphs (tDAGs), to model first-order rule feature spaces for sentence pair classification. We introduce a novel algorithm for computing the similarity in first-order rewrite rule feature spaces. Our algorithm is extremely efficient and, as it computes the similarity of instances that can be represented in explicit feature spaces, it is a valid kernel function.

1 Introduction

Natural language processing models are generally positive combinations between linguistic models and automatically learnt classifiers. As trees are extremely important in many linguistic theories, a large amount of works exploiting machine learning algorithms for NLP tasks has been developed for this class of data structures (Collins and Duffy, 2002; Moschitti, 2004). These works propose efficient algorithms for determining the similarity among two trees in tree fragment feature spaces.

Yet, some NLP tasks such as textual entailment recognition (Dagan and Glickman, 2004; Dagan et al., 2006) and some linguistic theories such as HPSG (Pollard and Sag, 1994) require more general graphs and, then, more general algorithms for computing similarity among graphs. Unfortunately, algorithms for computing similarity among two general graphs in term of common subgraphs are still exponential (Ramon and Gärtner, 2003). In these cases, approximated algorithms have been proposed. For example, the one proposed in (Gärtner, 2003) counts the number of subpaths in common. The same happens for the one proposed in (Suzuki et al., 2003) that is applicable to a particular class of graphs, i.e. the hierarchical directed acyclic graphs. These algorithms do not compute the number of subgraphs in common between two graphs. Then, these algorithms approximate the feature spaces we need in these NLP tasks. For computing similarities in these feature spaces, we have to investigate if we can define a particular class of graphs for the class of tasks we want to solve. Once we focused the class of graph, we can explore efficient similarity algorithms.

A very important class of graphs can be defined for tasks involving sentence pairs. In these cases, an important class of feature spaces is the one that represents first-order rewrite rules. For example, in textual entailment recognition (Dagan et al., 2006), we need to determine whether a text T implies a hypothesis H, e.g., whether or not "*Farmers feed cows animal extracts*" entails "*Cows eat animal extracts*" (T_1, H_1). If we want to learn textual entailment classifiers, we need to exploit first-order rules hidden in training instances. To positively exploit the training instance "*Pediatricians suggest women to feed newborns breast milk*" entails "*Pediatricians suggest that newborns eat breast milk*" (T_2, H_2) for classifying the above example, learning algorithms should learn that the two instances hide the first-order rule $\rho = feed\boxed{Y}\boxed{Z} \rightarrow \boxed{Y}eat\boxed{Z}$. The first-order rule feature space, introduced by (Zanzotto and Moschitti, 2006), gives high performances in term of accuracy for textual entailment recognition with respect to other features spaces.

In this paper, we propose a novel class of graphs, the tripartite directed acyclic graphs (tDAGs), that model first-order rule feature spaces and, using this class of graphs, we introduce a novel algorithm for computing the similarity in first-order rewrite rule feature spaces. The possibility of explicitly representing the first-order feature space as subgraphs of tDAGs makes the derived similarity function a valid kernel. With respect to the algorithm proposed in (Moschitti and Zanzotto, 2007), our algorithm is more efficient

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 91–100,
Singapore, 6-7 August 2009.

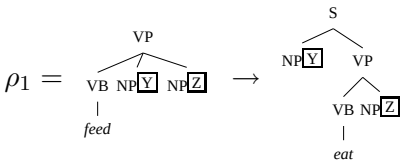
and it is a valid kernel function.

The paper is organized as follows. In Sec. 2, we firstly describe tripartite directed acyclic graphs (tDAGs) to model first-order feature (FOR) spaces. In Sec. 3, we then present the related work. In Sec. 4, we introduce the similarity function for these FOR spaces. This can be used as kernel function in kernel-based machines (e.g., support vector machines (Cortes and Vapnik, 1995)). We then introduce our efficient algorithm for computing the similarity among tDAGs. In Sec. 5, we analyze the computational efficiency of our algorithm showing that it is extremely more efficient than the algorithm proposed in (Moschitti and Zanzotto, 2007). Finally, in Sec. 6, we draw conclusions and plan the future work.

2 Representing first-order rules and sentence pairs as tripartite directed acyclic graphs

As first step, we want to define the *tripartite directed acyclic graphs* ($tDAGs$). This is an extremely important class of graphs for the first-order rule feature spaces we want to model. We want here to intuitively show that, if we model first-order rules and sentence pairs as $tDAG$s, determining whether or not a sentence pair can be unified with a first-order rewrite rule is a graph matching problem. This intuitive idea helps in determining our efficient algorithm for exploiting first-order rules in learning examples.

To illustrate the above idea we will use an example based on the above rule ρ= $feed$ Y Z $\rightarrow$ Y eat Z and the above sentence pair (T_1, H_1). The rule ρ encodes the entailment relation of the verb *to feed* and the verb *to eat*. If represented over a syntactic interpretation, the rule has the following aspect:



As in the case of feature structures (Carpenter, 1992), we can observe this rule as a graph. As we are not interested in the variable names but we need to know the relation between the right hand side and the left hand side of the rule, we can substitute each variable with an unlabelled node. We then connect tree nodes having variables with the corresponding unlabelled node. The result is a graph as the one in Fig. 1. The variables Y and Z are represented by the unlabelled nodes between the trees.

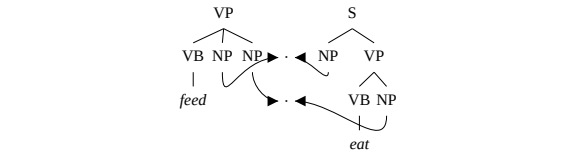


Figure 1: A simple rewrite rule seen as a graph

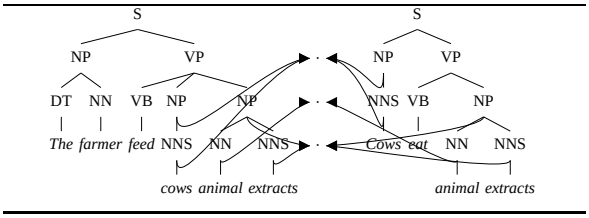


Figure 2: A sample pair seen as a graph

In the same way we can represent the sentence pair (T_1, H_1) using graph with explicit links between related words and nodes (see Fig. 2). We can link words using anchoring methods as in (Raina et al., 2005). These links can then be propagated in the syntactic tree using semantic heads of the constituents (Pollard and Sag, 1994). The rule ρ_1 matches over the pair (T_1, H_1) if the graph ρ_1 is among the subgraphs of the graph in Fig. 2.

Both rules and sentence pairs are graphs of the same type. These graphs are basically two trees connected through an intermediate set of nodes representing variables in the rules and relations between nodes in the sentence pairs. We will hereafter call these graphs *tripartite directed acyclic graphs* (tDAGs). The formal definition follows.

Definition tDAG: A *tripartite directed acyclic graph* is a graph $G = (N, E)$ where

- the set of nodes N is partitioned in three sets N_t, N_g, and A
- the set of edges is partitioned in four sets E_t, E_g, E_{A_t}, and E_{A_g}

such that $t = (N_t, E_t)$ and $g = (N_g, E_g)$ are two trees and $E_{A_t} = \{(x, y)|x \in N_t \text{ and } y \in A\}$ and $E_{A_g} = \{(x, y)|x \in N_g \text{ and } y \in A\}$ are the edges connecting the two trees.

A $tDAG$ is a partially labeled graph. The labeling function L only applies to the subsets of nodes related to the two trees, i.e., $L : N_t \cup N_g \rightarrow \mathcal{L}$. Nodes in the set A are not labeled.

The explicit representation of the tDAG in Fig. 2 has been useful to show that the unification of a rule and a sentence pair is a graph matching problem. Yet, it is complex to follow. We will then describe a tDAG with an alternative and more convenient representation. A tDAG $G = (N, E)$ can be seen as pair $G = (\tau, \gamma)$ of *extended trees* τ and γ where $\tau = (N_t \cup A, E_t \cup E_{A_t})$ and $\gamma = (N_g \cup A, E_g \cup E_{A_g})$. These are extended trees as each tree contains the relations with the other tree.

As for the feature structures, we will graphically represent a $(x, y) \in E_{A_t}$ and a $(z, y) \in E_{A_g}$ as boxes $\boxed{y}$ respectively on the node x and on the node z. These nodes will then appear as $L(x)\boxed{y}$ and $L(z)\boxed{y}$, e.g., NP$\boxed{1}$. The name y is not a label but a placeholder representing an unlabelled node. This representation is used for rules and for sentence pairs. The sentence pair in Fig. 2 is then represented as reported in Fig. 3.

3 Related work

Automatically learning classifiers for sentence pairs is extremely important for applications like textual entailment recognition, question answering, and machine translation.

In textual entailment recognition, it is not hard to see graphs similar to tripartite directed acyclic graphs as ways of extracting features from examples to feed automatic classifiers. Yet, these graphs are generally not tripartite in the sense described in the previous section and they are not used to extract features representing first-order rewrite rules. In (Raina et al., 2005; Haghighi et al., 2005; Hickl et al., 2006), two connected graphs representing the two sentences s_1 and s_2 are used to compute distance features, i.e., features representing the distance between s_1 and s_2. The underlying idea is that lexical, syntactic, and semantic similarities between sentences in a pair are relevant features to classify sentence pairs in classes such as *entail* and *not-entail*.

In (de Marneffe et al., 2006), first-order rewrite rule feature spaces have been explored. Yet, these spaces are extremely small. Only some features representing first-order rules have been explored. Pairs of graphs are used here to determine if a feature is active or not, i.e., the rule fires or not. A larger feature space of rewrite rules has been implicitly explored in (Wang and Neumann, 2007) but this work considers only ground rewrite rules.

In (Zanzotto and Moschitti, 2006), tripartite directed acyclic graphs are implicitly introduced and exploited to build first-order rule feature spaces. Yet, both in (Zanzotto and Moschitti, 2006) and in (Moschitti and Zanzotto, 2007), the model proposed has two major limitations: it can represent rules with less than 7 variables and the proposed kernel is not a completely valid kernel as it uses the max function.

In machine translation, some methods such as (Eisner, 2003) learn graph based rewrite rules for generative purposes. Yet, the method presented in (Eisner, 2003) can model first-order rewrite rules only with a very small amount of variables, i.e., two or three variables.

4 An efficient algorithm for computing the first-order rule space kernel

In this section, we present our idea for an efficient algorithm for exploiting first-order rule feature spaces. In Sec. 4.1, we firstly define the similarity function, i.e., the kernel $K(G_1, G_2)$, that we need to determine for correctly using first-order rules feature spaces. This kernel is strongly based on the isomorphism between graphs. A relevant idea of this paper is the observation that we can define an efficient way to detect the isomorphism between the tDAGs (Sec. 4.2). This algorithm exploits the efficient algorithms of tree isomorphism as the one implicitly used in (Collins and Duffy, 2002). After describing the isomorphism between tDAGs, We can present the idea of our efficient algorithm for computing $K(G_1, G_2)$ (Sec. 4.3). We introduce the algorithms to make it a viable solution (Sec. 4.4). Finally, in Sec. 4.5, we report the kernel computation we compare against presented by (Zanzotto and Moschitti, 2006; Moschitti and Zanzotto, 2007).

4.1 Kernel functions over first-order rule feature spaces

The first-order rule feature space we want to model is huge. If we use kernel-based machine learning models such as SVM (Cortes and Vapnik, 1995), we can implicitly define the space by defining its similarity functions, i.e., its kernel functions. We firstly introduce the first-order rule feature space and we then define the prototypical kernel function over this space.

The first-order rule feature space (FOR) is in general the space of all the possible first-order

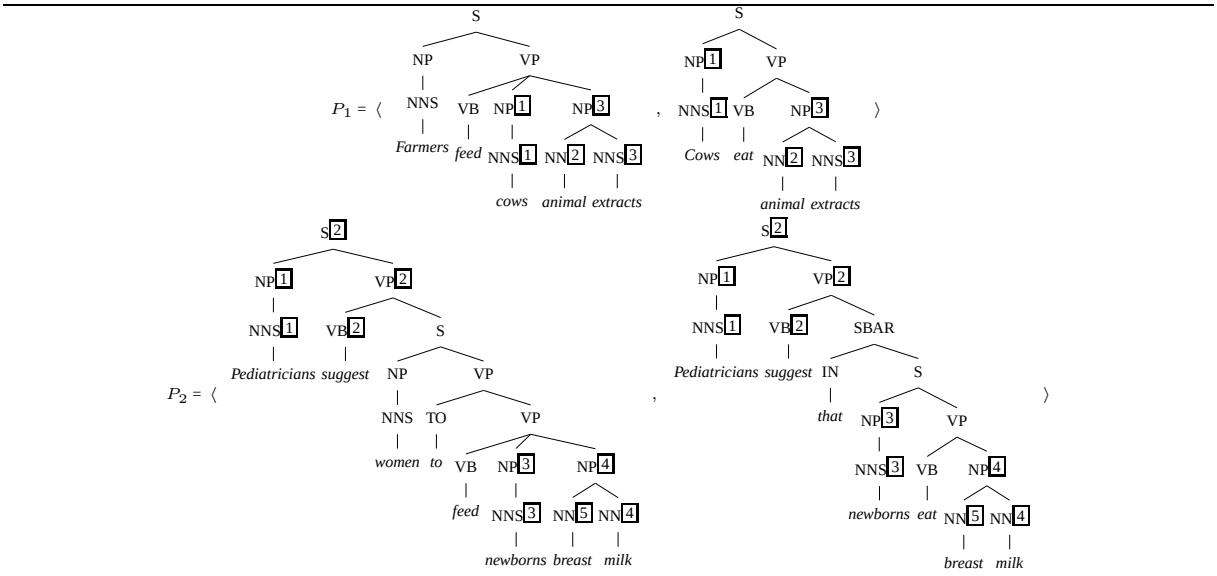
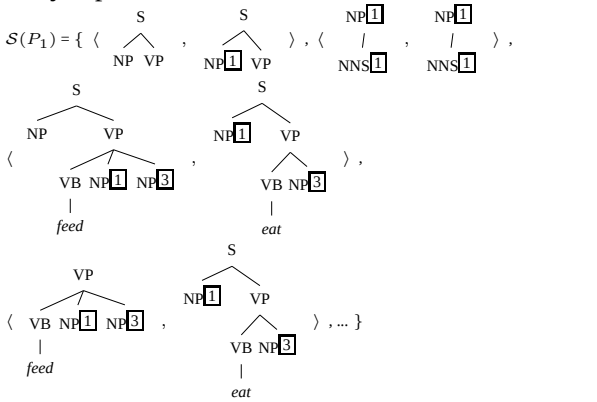
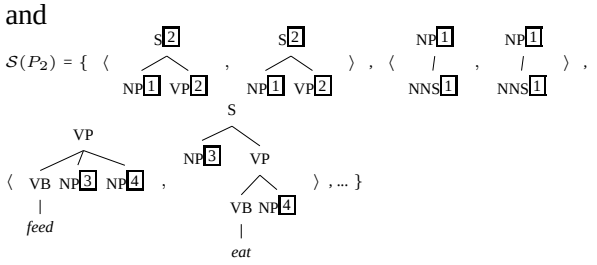


Figure 3: Two tripartite DAGs

rules defined as tDAGs. Within this space it is possible to define the function $\mathcal{S}(G)$ that determines all the possible active features of the tDAG G in FOR. The function $\mathcal{S}(G)$ determines all the possible and meaningful subgraphs of G. We want that these subgraphs represent first-order rules that can be matched with the pair G. Then, meaningful subgraphs of $G = (\tau, \gamma)$ are graphs as (t, g) where t and g are subtrees of τ and γ. For example, the subgraphs of P_1 and P_2 in Fig. 3 are hereafter partially represented:



and



In the FOR space, the kernel function K should then compute the number of subgraphs in common. The trivial way to describe the former kernel function is using the intersection operator, i.e., the kernel $K(G_1, G_2)$ is the following:

$$K(G_1, G_2) = |\mathcal{S}(G_1) \cap \mathcal{S}(G_2)| \qquad (1)$$

This is very simple to write and it is in principle correct. A graph g in the intersection $\mathcal{S}(G_1) \cap \mathcal{S}(G_2)$ is a graph that belongs to both $\mathcal{S}(G_1)$ and $\mathcal{S}(G_2)$. Yet, this hides a very important fact: determining whether two graphs, g_1 and g_2, are the *same* graph $g_1 = g_2$ is not trivial. For example, it is not sufficient to superficially compare graphs to determine that ρ_1 belongs both to $\mathcal{S}_1$ and $\mathcal{S}_2$. We need to use the correct property for $g_1 = g_2$, i.e., the *isomorphism* between two graphs. We can call the operator $Iso(g_1, g_2)$. When two graphs verify the property $Iso(g_1, g_2)$, both g_1 and g_2 can be taken as the graph g representing the two graphs. Detecting $Iso(g_1, g_2)$ has an exponential complexity (Köbler et al., 1993).

This complexity of the intersection operator between sets of graphs deserves a different way to represent the operation. We will use the same symbol but we will use the prefix notation. The operator is hereafter re-defined:

$$\cap(\mathcal{S}(G_1), \mathcal{S}(G_2)) =$$
$$= \{g_1 | g_1 \in \mathcal{S}(G_1), \exists g_2 \in \mathcal{S}(G_2), Iso(g_1, g_2)\}$$

4.2 Isomorphism between tDAGs

As isomorphism between graphs is an essential activity for learning from structured data, we here review its definition and we adapt it to tDAGs.

We then observe that isomorphism between two tDAGs can be divided in two sub-problems:

- finding the isomorphism between two pairs of *extended trees*
- checking whether the partial isomorphism found between the two pairs of *extended trees* are compatible.

In general, two tDAGs, $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$ are *isomorphic* (or match) if $|N_1| = |N_2|$, $|E_1| = |E_2|$, and a bijective function $f : N_1 \rightarrow N_2$ exists such that these properties hold:

- for each node $n \in N_1$, $L(f(n)) = L(n)$
- for each edge $(n_1, n_2) \in E_1$ an edge $(f(n_1), f(n_2))$ is in E_2

The bijective function f is a member of the combinatorial set $\mathcal{F}$ of all the possible bijective functions between the two sets N_1 and N_2.

The trivial algorithm for detecting if two graphs are isomorphic is exponential (Köbler et al., 1993). It explores all the set $\mathcal{F}$. It is still undetermined if the general graph isomorphism problem is NP-complete. Yet, we can use the fact that tDAGs are two extended trees for building a better algorithm. There is an efficient algorithm for computing isomorphism between trees (as the one implicitly used in (Collins and Duffy, 2002)).

Given two tDAGs $G_1 = (\tau_1, \gamma_1)$ and $G_2 = (\tau_2, \gamma_2)$ the isomorphism problem can be divided in detecting two properties:

1. *Partial isomorphism*. Two tDAGs G_1 and G_2 are *partially isomorphic*, if τ_1 and τ_2 are isomorphic and if γ_1 and γ_2 are isomorphic. The partial isomorphism produces two bijective functions f_τ and f_γ.
2. *Constraint compatibility*. Two bijective functions f_τ and f_γ are compatible on the sets of nodes A_1 and A_2, if for each $n \in A_1$, it happens that $f_\tau(n) = f_\gamma(n)$.

We can rephrase the second property, i.e., the constraint compatibility, as follows. We define two constraints $c(\tau_1, \tau_2)$ and $c(\gamma_1, \gamma_2)$ representing the functions f_τ and f_γ on the sets A_1 and A_2. The two constraints are defined as $c(\tau_1, \tau_2) = \{(n, f_\tau(n)) | n \in A_1\}$ and $c(\gamma_1, \gamma_2) = \{(n, f_\gamma(n)) | n \in A_1\}$. Two partially isomorphic tDAGs are isomorphic if the constraints match, i.e., $c(\tau_1, \tau_2) = c(\gamma_1, \gamma_2)$.

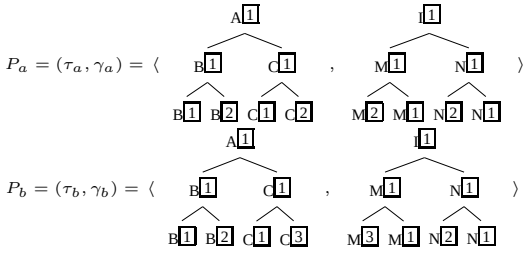


Figure 5: Simple non-linguistic tDAGs

For example, the third pair of $\mathcal{S}(P_1)$ and the second pair of $\mathcal{S}(P_2)$ are isomorphic as: (1) these are partially isomorphic, i.e., the right hand sides τ and the left hand sides γ are isomorphic; (2) both pairs of extended trees generate the constraint $c_1 = \{(\boxed{1}, \boxed{3}), (\boxed{3}, \boxed{4})\}$. In the same way, the fourth pair of $\mathcal{S}(P_1)$ and the third pair of $\mathcal{S}(P_2)$ generate $c_2 = \{(\boxed{1}, \boxed{1})\}$

4.3 General idea for an efficient kernel function

As above discussed, two tDAGs are isomorphic if the two properties, the *partial isomorphism* and the *constraint compatibility*, hold. To compute the kernel function $K(G_1, G_2)$ defined in Sec. 4.1, we can exploit these properties in the reverse order. Given a constraint c, we can select all the graphs that meet the constraint c (*constraint compatibility*). Having the two set of all the tDAGs meeting the constraint, we can detect the *partial isomorphism*. We split each pair of tDAGs in the four extended trees and we determine if these extended trees are compatible.

We introduce this innovative method to compute the kernel $K(G_1, G_2)$ in the FOR space in two steps. Firstly, we give an intuitive explanation and, secondly, we formally define the kernel.

4.3.1 Intuitive explanation

To give an intuition of the kernel computation, without loss of generality and for sake of simplicity, we use two non-linguistic tDAGs, P_a and P_b (see Fig. 5), and the subgraph function $\widetilde{\mathcal{S}}(\theta)$. This latter is an approximated version of $\mathcal{S}(\theta)$ that generates tDAGs with subtrees rooted in the root of the initial trees of θ.

To exploit the *constraint compatibility* property, we define C as *the set of all the relevant alternative constraints*, i.e., the constraints c that are likely to be generated when detecting the *partial isomorphism*. For P_a and P_b, this set is $C = \{c_1, c_2\} =$

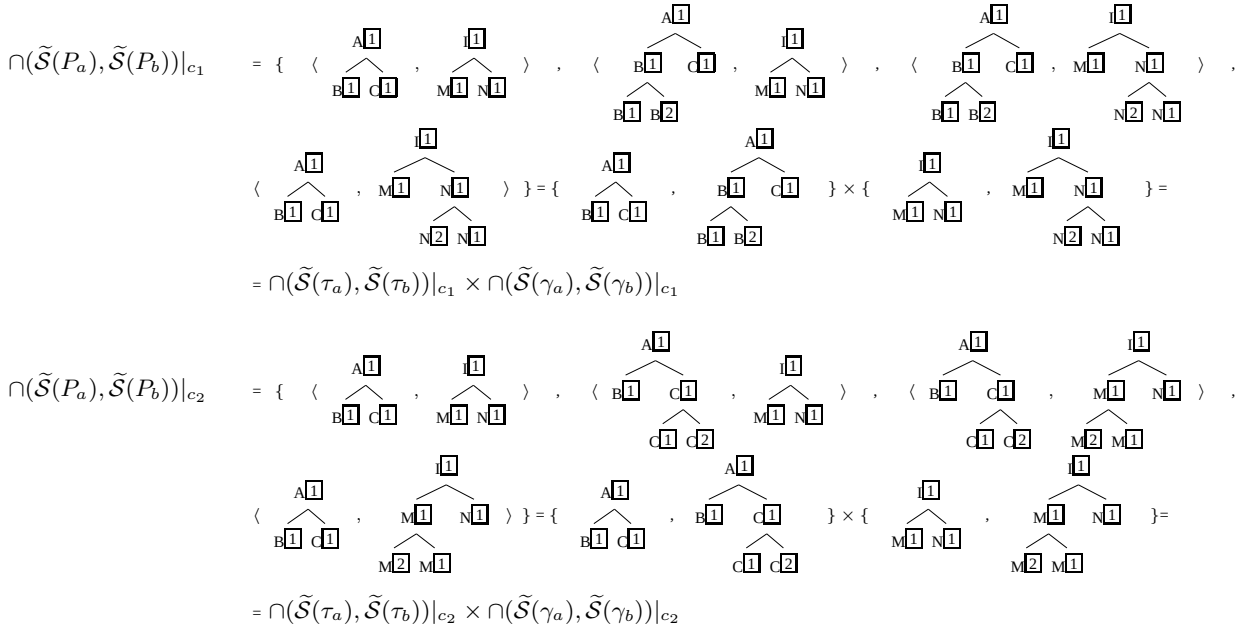


Figure 4: Intuitive idea for the kernel computation

$\{\{(\boxed{1},\boxed{1}),(\boxed{2},\boxed{2})\},\{(\boxed{1},\boxed{1}),(\boxed{2},\boxed{3})\}\}$. We can then determine the kernel $K(P_a, P_b)$ as:

$$K(P_a,P_b) = |\cap(\widetilde{\mathcal{S}}(P_a),\widetilde{\mathcal{S}}(P_b))| = \left|\cap(\widetilde{\mathcal{S}}(P_a),\widetilde{\mathcal{S}}(P_b))|_{c_1} \bigcup \cap(\widetilde{\mathcal{S}}(P_a),\widetilde{\mathcal{S}}(P_b))|_{c_2}\right|$$

where $\cap(\widetilde{\mathcal{S}}(P_a),\widetilde{\mathcal{S}}(P_b))|_c$ are the common subgraphs that meet the constraint c. A tDAG $g' = (\tau', \gamma')$ in $\widetilde{\mathcal{S}}(P_a)$ is in $\cap(\widetilde{\mathcal{S}}(P_a),\widetilde{\mathcal{S}}(P_b))|_c$ if $g'' = (\tau'', \gamma'')$ in $\widetilde{\mathcal{S}}(P_b)$ exists, g' is partially isomorphic to g'', and $c' = c(\tau', \tau'') = c(\gamma', \gamma'')$ is *covered* by and *compatible* with the constraint c, i.e., $c' \subseteq c$. For example in Fig. 4, the first tDAG of the set $\cap(\widetilde{\mathcal{S}}(P_a),\widetilde{\mathcal{S}}(P_b))|_{c_1}$ belongs to the set as its constraint $c' = \{(\boxed{1},\boxed{1})\}$ is a subset of c_1.

Observing the kernel computation in this way is important. Elements in $\cap(\widetilde{\mathcal{S}}(P_a),\widetilde{\mathcal{S}}(P_b))|_c$ already satisfy the property of *constraint compatibility*. We only need to determine if the *partially isomorphic* properties hold for elements in $\cap(\widetilde{\mathcal{S}}(P_a),\widetilde{\mathcal{S}}(P_b))|_c$. Then, we can write the following equivalence:

$$\cap(\widetilde{\mathcal{S}}(P_a),\widetilde{\mathcal{S}}(P_b))|_c = \cap(\widetilde{\mathcal{S}}(\tau_a),\widetilde{\mathcal{S}}(\tau_b))|_c \times \cap(\widetilde{\mathcal{S}}(\gamma_a),\widetilde{\mathcal{S}}(\gamma_b))|_c \quad (2)$$

Figure 4 reports this equivalence for the two sets derived using the constraints c_1 and c_2. Note that this equivalence is not valid if a constraint is not applied, i.e., $\cap(\widetilde{\mathcal{S}}(P_a),\widetilde{\mathcal{S}}(P_b)) \neq \cap(\widetilde{\mathcal{S}}(\tau_a),\widetilde{\mathcal{S}}(\tau_b)) \times \cap(\widetilde{\mathcal{S}}(\gamma_a),\widetilde{\mathcal{S}}(\gamma_b))$. The pair P_a itself does not belong to $\cap(\widetilde{\mathcal{S}}(P_a),\widetilde{\mathcal{S}}(P_b))$ but it does belong to $\cap(\widetilde{\mathcal{S}}(\tau_a),\widetilde{\mathcal{S}}(\tau_b)) \times \cap(\widetilde{\mathcal{S}}(\gamma_a),\widetilde{\mathcal{S}}(\gamma_b))$.

The equivalence (2) allows to compute the cardinality of $\cap(\widetilde{\mathcal{S}}(P_a),\widetilde{\mathcal{S}}(P_b))|_c$ using the cardinalities of $\cap(\widetilde{\mathcal{S}}(\tau_a),\widetilde{\mathcal{S}}(\tau_b))|_c$ and $\cap(\widetilde{\mathcal{S}}(\gamma_a),\widetilde{\mathcal{S}}(\gamma_b))|_c$. These latter sets contain only extended trees where the equivalences between unlabelled nodes are given by c. We can then compute the cardinalities of these two sets using methods developed for trees (e.g., the kernel function $K_S(\theta_1, \theta_2)$ introduced in (Collins and Duffy, 2002)).

4.3.2 Formal definition

Given the idea of the previous section, it is easy to demonstrate that the kernel $K(G_1, G_2)$ can be written as follows:

$$K(G_1,G_2) = \left|\bigcup_{c\in C} \cap(\mathcal{S}(\tau_1),\mathcal{S}(\tau_2))|_c \times \cap(\mathcal{S}(\gamma_1),\mathcal{S}(\gamma_2))|_c\right|$$

where C is set of alternative constraints and $\cap(\mathcal{S}(\theta_1),\mathcal{S}(\theta_2))|_c$ are all the common extended trees compatible with the constraint c.

We can compute the above kernel using the inclusion-exclusion property, i.e.,

$$|A_1 \cup \cdots \cup A_n| = \sum_{J\in 2^{\{1,\ldots,n\}}} (-1)^{|J|-1}|A_J| \quad (3)$$

where $2^{\{1,\ldots,n\}}$ is the set of all the subsets of $\{1, \ldots, n\}$ and $A_J = \bigcap_{i\in J} A_i$.

To describe the application of the inclusion-exclusion model in our case, let firstly define:

$$K_S(\theta_1, \theta_2, c) = |\cap(\mathcal{S}(\theta_1),\mathcal{S}(\theta_2))|_c| \quad (4)$$

where θ_1 can be both τ_1 and γ_1 and θ_2 can be both τ_2 and γ_2. Trivially, we can demonstrate that:

$$K(G_1, G_2) = \\ = \sum_{J \in 2^{\{1,\ldots,|C|\}}} (-1)^{|J|-1} K_S(\tau_1, \tau_2, c(J)) K_S(\gamma_1, \gamma_2, c(J)) \quad (5)$$

where $c(J) = \bigcap_{i \in J} c_i$.

Given the nature of the constraint set C, we can compute efficiently the previous equation as it often happens that two different J_1 and J_2 in $2^{\{1,\ldots,|C|\}}$ generate the same c, i.e.

$$c = \bigcap_{i \in J_1} c_i = \bigcap_{i \in J_2} c_i \quad (6)$$

Then, we can define C^* as the set of all intersections of constraints in C, i.e. $C^* = \{c(J) | J \in 2^{\{1,\ldots,|C|\}}\}$. We can rewrite the equation as:

$$K(G_1, G_2) = \\ = \sum_{c \in C^*} K_S(\tau_1, \tau_2, c) K_S(\gamma_1, \gamma_2, c) N(c) \quad (7)$$

where

$$N(c) = \sum_{\substack{J \in 2^{\{1,\ldots,|C|\}} \\ c = c(J)}} (-1)^{|J|-1} \quad (8)$$

The complexity of the above kernel strongly depends on the cardinality of C and the related cardinality of C^*. The worst-case computational complexity is still exponential with respect to the size of A_1 and A_2. Yet, the average case complexity (Wang, 1997) is promising.

The set C is generally very small with respect to the worst case. If $\mathcal{F}_{(A_1,A_2)}$ are all the possible correspondences between the nodes A_1 and A_2, it happens that $|C| << |\mathcal{F}_{(A_1,A_2)}|$ where $|\mathcal{F}_{(A_1,A_2)}|$ is the worst case. For example, in the case of P_1 and P_2, the cardinality of $C = \{\{(1,1)\}, \{(1,3),(3,4),(2,5)\}\}$ is extremely smaller than the one of $\mathcal{F}_{(A_1,A_2)} = \{\{(1,1),(2,2),(3,3)\}, \{(1,2),(2,1),(3,3)\}, \{(1,2),(2,3),(3,1)\}, ..., \{(1,3),(2,4),(3,5)\}\}$. In Sec. 4.5 we argue that the algorithm presented in (Moschitti and Zanzotto, 2007) has the worst-case complexity.

Moreover, the set C^* is extremely smaller than $2^{\{1,\ldots,|C|\}}$ due to the above property (6).

We will analyze the average-case complexity with respect to the worst-case complexity in Sec. 5.

4.4 Enabling the efficient kernel function

The above idea for computing the kernel function is extremely interesting. Yet, we need to make it viable by describing the way we can determine efficiently the three main parts of the equation (7): 1) the set of alternative constraints C (Sec. 4.4.1); 2) the set C^* of all the possible intersections of constraints in C (Sec. 4.4.2); and, finally, 3) the numbers $N(c)$ (Sec. 4.4.3).

4.4.1 Determining the set of alternative constraints

The first step of equation (7) is to determine the alternative constraints C. We can here strongly use the possibility of dividing tDAGs in two trees. We build C as $C_\tau \cup C_\gamma$ where: 1) C_τ are the constraints obtained from pairs of isomorphic extended trees $t_1 \in \mathcal{S}(\tau_1)$ and $t_2 \in \mathcal{S}(\tau_2)$; 2) C_γ are the constraints obtained from pairs of isomorphic extended trees $t_1 \in \mathcal{S}(\gamma_1)$ and $t_2 \in \mathcal{S}(\gamma_2)$.

The idea for an efficient algorithm is that we can compute the C without explicitly looking at all the subgraphs involved. We instead use and combine the constraints derived comparing the productions of the extended trees. We can compute then C_τ with the productions of τ_1 and τ_2 and C_γ with the productions of γ_1 and γ_2. For example (see Fig. 3), focusing on the τ, the rule $NP\boxed{3} \rightarrow NN\boxed{2}NNS\boxed{3}$ of G_1 and $NP\boxed{4} \rightarrow NN\boxed{5}NNS\boxed{4}$ of G_2 generates the constraint $c = \{(\boxed{3},\boxed{4}),(\boxed{2},\boxed{5})\}$.

Using the above intuition it is possible to define an algorithm that builds an alternative constraint set C with the following two properties:

1. for each common subtree according to a set of constraints c, $\exists c' \in C$ such that $c \subseteq c'$;
2. $\nexists c', c'' \in C$ such that $c' \subset c''$ and $c' \neq \emptyset$.

4.4.2 Determining the set C^*

The set C^* is defined as the set of all possible intersections of alternative constraints in C. Figure 6 presents the algorithm determining C^*. Due to the property (6) discussed in Sec. 4.3, we can empirically demonstrate that the average complexity of the algorithm is not bigger than $O(|C|^2)$. Yet, again, the worst case complexity is exponential.

4.4.3 Determining the values of $N(c)$

The multiplier $N(c)$ (Eq. 8) represents the number of times the constraint c is considered in the sum of equation 5, keeping into account the sign of

Algorithm *Build the set C^* from the set C*

$C^+ \leftarrow C$; $C_1 \leftarrow C$; $C_2 \leftarrow \emptyset$
WHILE $|C_1| > 1$
  FORALL $c' \in C_1$
    FORALL $c'' \in C_1$ such that $c' \neq c''$
      $c \leftarrow c' \cap c''$
      IF $c \notin C^+$ add c to C_2
  $C^+ \leftarrow C^+ \cup C_2$; $C_1 \leftarrow C_2$; $C_2 \leftarrow \emptyset$
$C^* \leftarrow C \cup C^+ \cup \{\emptyset\}$

Figure 6: Algorithm for computing C^*

the corresponding addend. It is possible to demonstrate that:

$$N(c) = 1 - \sum_{\substack{c' \in C^* \\ c' \supset c}} N_{c'} \quad (9)$$

This recursive formulation of the equation allows us to easily determine the value of $N(c)$ for every c belonging to C^*. It is possible to prove this property using set properties and the binomial theorem. The proof is omitted for lack of space.

4.5 Reviewing the strictly related work

To understand if ours is an efficient algorithm, we compare it with the algorithm presented by (Moschitti and Zanzotto, 2007). We will hereafter call this algorithm K_{max}. The K_{max} algorithm and kernel is an approximation of what is a kernel needed for a FOR space as it is not difficult to demonstrate that $K_{max}(G_1, G_2) \leq K(G_1, G_2)$. The K_{max} approximation is based on maximization over the set of possible correspondences of the placeholders. Following our formulation, this kernel appears as:

$$K_{max}(G_1, G_2) = \\ = \max_{c \in \mathcal{F}_{(A_1, A_2)}} K_S(\tau_1, \tau_2, c) K_S(\gamma_1, \gamma_2, c) \quad (10)$$

where $\mathcal{F}_{(A_1,A_2)}$ are all the possible correspondences between the nodes A_1 and A_2 of the two tDAGs as the one presented in Sec. 4.3. This formulation of the kernel has the worst case complexity of our formulation, i.e., Eq. 7.

For computing the basic kernel for the extended trees, i.e. $K_S(\theta_1, \theta_2, c)$ we use the model algorithm presented by (Zanzotto and Moschitti, 2006) and refined by (Moschitti and Zanzotto, 2007) based on the algorithm for tree fragment feature spaces (Collins and Duffy, 2002). As we are using the same basic kernel, we can empirically compare the two methods.

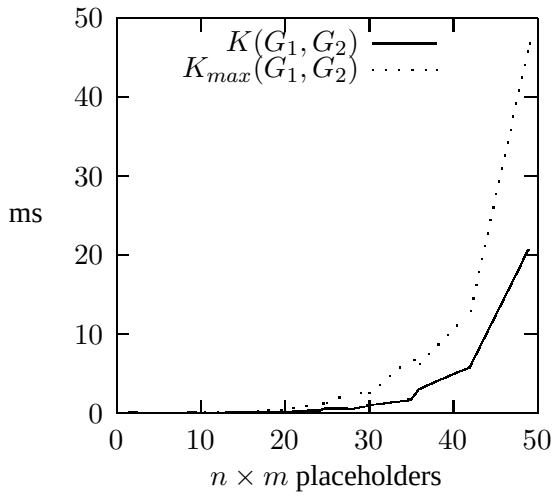


Figure 7: Mean execution time in milliseconds (ms) of the two algorithms wrt. $n \times m$ where n and m are the number of placeholders of the two tDAGs

5 Experimental evaluation

In this section we want to empirically estimate the benefits on the computational cost of our novel algorithm with respect to the algorithm proposed by (Moschitti and Zanzotto, 2007). Our algorithm is in principle exponential with respect to the set of alternative constraints C. Yet, due to what presented in Sec. 4.4 and as the set C^* is usually very small, the average complexity is extremely low. Following the theory on the average-cost computational complexity (Wang, 1997), we estimated the behavior of the algorithms on a large distribution of cases. We then compared the computing times of the two algorithms. Finally, as K and K_{max} compute slightly different kernels, we compare the accuracy of the two methods. We implemented both algorithms $K(G_1, G_2)$ and $K_{max}(G_1, G_2)$ in support vector machine classifier (Joachims, 1999) and we experimented with both implementations on the same machine. We hereafter analyze the results in term of execution time (Sec. 5.1) and in term of accuracy (Sec. 5.2).

5.1 Average computing time analysis

For this first set of experiments, the source of examples is the one of the recognizing textual entailment challenge, i.e., RTE2 (Bar-Haim et al.,

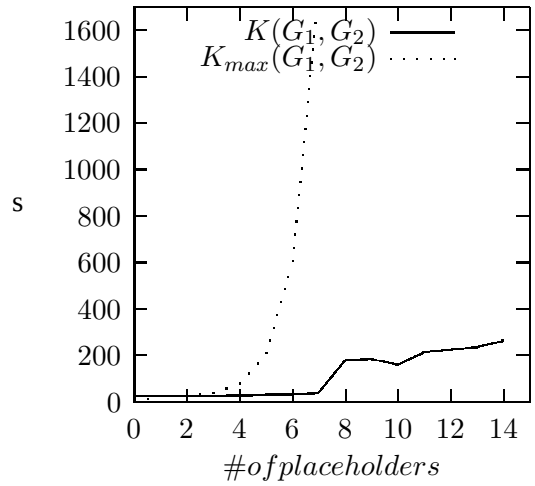


Figure 8: Total execution time in seconds (s) of the training phase on RTE2 wrt. different numbers of allowed placeholders

2006). The dataset of the challenge has 1,600 sentence pairs.

The computational cost of both $K(G_1, G_2)$ and $K_{max}(G_1, G_2)$ depends on the number of placeholders $n = |A_1|$ of G_1 and on $m = |A_2|$ the number of placeholders of G_2. Then, in the first experiment we want to determine the relation between the computational time and the factor $n \times m$. Results are reported in Fig. 7 where the computation times are plotted with respect to $n \times m$. Each point in the curve represents the average execution time for the pairs of instances having $n \times m$ placeholders. As expected, the computation of the function K is more efficient than the computation K_{max}. The difference between the two execution times increases with $n \times m$.

We then performed a second experiment that wants to determine the relation of the total execution with the maximum number of placeholders in the examples. This is useful to estimate the behavior of the algorithm with respect to its application in learning models. Using the RTE2 data, we artificially build different versions with increasing number of placeholders. We then have RTE2 with 1 placeholder at most in each pair, RTE2 with 2 placeholders, etc. The number of pairs in each set is the same. What changes is the maximal number of placeholders. Results are reported in Fig. 8 where the execution time of the training phase in seconds (s) is plotted for each different set. We see that the computation of K_{max} is exponential with respect to the number of placeholders and it becomes intractable after 7 placeholders. The computation of K is instead more flat. This can be explained as the computation of K is related to the real alternative constraints that appears in the dataset. The computation of the kernel K then outperforms the computation of the kernel K_{max}.

Kernel	Accuracy	Used training examples	Support Vectors
K_{max}	59.32	4223	4206
K	60.04	4567	4544

Table 1: Comparative performances of K_{max} and K

5.2 Accuracy analysis

As K_{max} that has been demonstrated very effective in term of accuracy for RTE and K compute a slightly different similarity function, we want to show that the performance of our more computationally efficient K is comparable, and even better, to the performances of K_{max}. We then performed an experiment taking as training all the data derived from RTE1, RTE2, and RTE3, (i.e., 4567 training examples) and taking as testing RTE-4 (i.e., 1000 testing examples). The results are reported in Tab. 1. As the table shows, the accuracy of K is higher than the accuracy of K_{max}. There are two main reasons. The first is that K_{max} is an approximation of K. The second is that we can now consider sentence pairs with more than 7 placeholders. Then, we can use the complete training set as the third column of the table shows.

6 Conclusions and future work

We presented an interpretation of first order rule feature spaces as *tripartite directed acyclic graphs* ($tDAG$s). This view on the problem gave us the possibility of defining a novel and efficient algorithm for computing the kernel function for first order rule feature spaces. Moreover, the resulting algorithm is a valid kernel as it can be written as dot product in the explicit space of the tDAG fragments. We demonstrated that our algorithm outperforms in term of average complexity the previous algorithm and it yields to better accuracies for the final task. We are investigating if this is a valid algorithm for two general directed acyclic graphs.

References

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, and Idan Magnini, Bernardo Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*. Venice, Italy.

Bob Carpenter. 1992. *The Logic of Typed Feature Structures*. Cambridge University Press, Cambridge, England.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL02*.

C. Cortes and V. Vapnik. 1995. Support vector networks. *Machine Learning*, 20:1–25.

Ido Dagan and Oren Glickman. 2004. Probabilistic textual entailment: Generic applied modeling of language variability. In *Proceedings of the Workshop on Learning Methods for Text Understanding and Mining*, Grenoble, France.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In Quionero-Candela et al., editor, *LNAI 3944: MLCW 2005*, pages 177–190, Milan, Italy. Springer-Verlag.

Marie-Catherine de Marneffe, Bill MacCartney, Trond Grenager, Daniel Cer, Anna Rafferty, and Christopher D. Manning. 2006. Learning to distinguish valid textual entailments. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, Venice, Italy.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), Companion Volume*, pages 205–208, Sapporo, July.

Thomas Gärtner. 2003. A survey of kernels for structured data. *SIGKDD Explorations*.

Aria D. Haghighi, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via graph matching. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 387–394, Morristown, NJ, USA. Association for Computational Linguistics.

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing textual entailment with LCCs GROUNDHOG system. In Bernardo Magnini and Ido Dagan, editors, *Proceedings of the Second PASCAL Recognizing Textual Entailment Challenge*, Venice, Italy. Springer-Verlag.

Thorsten Joachims. 1999. Making large-scale svm learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods-Support Vector Learning*. MIT Press.

Johannes Köbler, Uwe Schöning, and Jacobo Torán. 1993. *The graph isomorphism problem: its structural complexity*. Birkhauser Verlag, Basel, Switzerland, Switzerland.

Alessandro Moschitti and Fabio Massimo Zanzotto. 2007. Fast and effective kernels for relational learning from texts. In *Proceedings of the International Conference of Machine Learning (ICML)*. Corvallis, Oregon.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *proceedings of the ACL*, Barcelona, Spain.

C. Pollard and I.A. Sag. 1994. *Head-driven Phrase Structured Grammar*. Chicago CSLI, Stanford.

Rajat Raina, Aria Haghighi, Christopher Cox, Jenny Finkel, Jeff Michels, Kristina Toutanova, Bill MacCartney, Marie-Catherine de Marneffe, Manning Christopher, and Andrew Y. Ng. 2005. Robust textual inference using diverse knowledge sources. In *Proceedings of the 1st Pascal Challenge Workshop*, Southampton, UK.

Jan Ramon and Thomas Gärtner. 2003. Expressivity versus efficiency of graph kernels. In *First International Workshop on Mining Graphs, Trees and Sequences*.

Jun Suzuki, Tsutomu Hirao, Yutaka Sasaki, and Eisaku Maeda. 2003. Hierarchical directed acyclic graph kernel: Methods for structured natural language data. In *In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 32–39.

Rui Wang and Günter Neumann. 2007. Recognizing textual entailment using a subsequence kernel method. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-07), July 22-26, Vancouver, Canada*.

Jie Wang. 1997. Average-case computational complexity theory. pages 295–328.

Fabio Massimo Zanzotto and Alessandro Moschitti. 2006. Automatic learning of textual entailments with cross-pair similarities. In *Proceedings of the 21st Coling and 44th ACL*, pages 401–408. Sydney, Australia, July.

Graphical Models over Multiple Strings*

Markus Dreyer and **Jason Eisner**
Department of Computer Science / Johns Hopkins University
Baltimore, MD 21218, USA
{markus,jason}@cs.jhu.edu

Abstract

We study graphical modeling in the case of *string-valued* random variables. Whereas a weighted finite-state transducer can model the probabilistic relationship between *two* strings, we are interested in building up joint models of *three or more strings*. This is needed for inflectional paradigms in morphology, cognate modeling or language reconstruction, and multiple-string alignment. We propose a Markov Random Field in which each factor (potential function) is a weighted finite-state machine, typically a transducer that evaluates the relationship between just two of the strings. The full joint distribution is then a product of these factors. Though decoding is actually undecidable in general, we can still do efficient joint inference using approximate belief propagation; the necessary computations and messages are all finite-state. We demonstrate the methods by jointly predicting morphological forms.

1 Overview

This paper considers what happens if a graphical model's variables can range over *strings* of unbounded length, rather than over the typical *finite* domains such as booleans, words, or tags. Variables that are connected in the graphical model are related by some weighted finite-state transduction.

Graphical models have become popular in machine learning as a principled way to work with collections of interrelated random variables. Most often they are used as follows:

1. **Build:** Manually specify the n variables of interest; their domains; and the possible direct interactions among them.
2. **Train:** Train this model's parameters θ to obtain a specific joint probability distribution $p(V_1, \ldots, V_n)$ over the n variables.
3. **Infer:** Use this joint distribution to predict the values of various unobserved variables from observed ones.

*Supported by the Human Language Technology Center of Excellence at Johns Hopkins University, and by National Science Foundation grant No. 0347822 to the second author.

Note that 1. requires intuitions about the domain; 2. requires some choice of training procedure; and 3. requires a choice of exact or approximate inference algorithm.

Our graphical models over strings are natural objects to investigate. We motivate them with some natural applications in computational linguistics (section 2). We then give our formalism: a Markov Random Field whose potential functions are rational weighted languages and relations (section 3). Next, we point out that inference is in general undecidable, and explain how to do approximate inference using message-passing algorithms such as belief propagation (section 4). The messages are represented as weighted finite-state machines.

Finally, we report on some initial experiments using these methods (section 7). We use incomplete data to train a joint model of morphological paradigms, then use the trained model to complete the data by predicting unseen forms.

2 Motivation

The problem of mapping between different forms and representations of strings is ubiquitous in natural language processing and computational linguistics. This is typically done between string *pairs*, where a pronunciation is mapped to its spelling, an inflected form to its lemma, a spelling variant to its canonical spelling, or a name is transliterated from one alphabet into another.
However, many problems involve more than just two strings:

- in *morphology*, the inflected forms of a (possibly irregular) verb are naturally considered together as a whole morphological paradigm in which different forms reinforce one another;
- mapping an English word to its foreign *transliteration* may be easier when one considers the orthographic *and* phonological forms of both words;
- similar *cognates* in multiple languages are naturally described together, in orthographic or phonological representations, or both;

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 101–110,
Singapore, 6-7 August 2009.

- modern and ancestral word forms form a phylogenetic tree in *historical linguistics*;
- in *bioinformatics* and in *system combination*, multiple sequences need to be aligned in order to identify regions of similarity.

We propose a unified model for multiple strings that is suitable for all the problems mentioned above. It is robust and configurable and can make use of task-specific overlapping features. It learns from observed and unobserved, or latent, information, making it useful in supervised, semi-supervised, and unsupervised settings.

3 Formal Modeling Approach

3.1 Variables

A **Markov Random Field** (MRF) is a joint model of a set of random variables, $\mathcal{V} = \{V_1, \ldots, V_n\}$. We assume that all variables are string-valued, i.e. the value of V_i may be any string $\in \Sigma_i^*$, where Σ_i is some finite alphabet.

We may use meaningful names for the integers i, such as $V_{2\mathrm{SA}}$ for the ***2nd singular past*** form of a verb.

The assumption that *all* variables are string-valued is not crucial; it merely simplifies our presentation. It is, however, sufficient for many practical purposes, since most other discrete objects can be easily encoded as strings. For example, if V_1 is a part of speech tag, it may be encoded as a length-1 string over the finite alphabet $\Sigma_1 \stackrel{\text{def}}{=} \{\mathsf{Noun}, \mathsf{Verb}, \ldots\}$.

3.2 Factors

A Markov Random Field defines a probability for each assignment $\mathcal{A}$ of values to the variables in $\mathcal{V}$:

$$p(\mathcal{A}) \stackrel{\text{def}}{=} \frac{1}{Z} \prod_{j=1}^{m} F_j(\mathcal{A}) \qquad (1)$$

This distribution over assignments is specified by the collection of **factors** $F_j : \mathcal{A} \mapsto \mathbb{R}_{\geq 0}$. Each factor (or **potential function**) is a function that depends on only a *subset* of $\mathcal{A}$.

Fig. 1 displays an undirected **factor graph**, in which each factor is connected to the variables that it depends on. F_1, F_3, F_5 in this example are **unary** factors because each one scores the value of a single variable, while F_2, F_4, F_6 are **binary** factors.

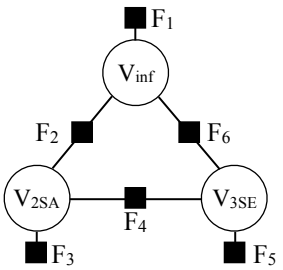


Figure 1: Example of a factor graph. Black boxes represent factors, circles represent variables (infinitive, 2nd past, and 3rd present-tense forms of the same verb; different samples from the MRF correspond to different verbs). Binary factors evaluate how well one string can be transduced into another, summing over all transducer paths (i.e., alignments, which are not observed in training).

In our setting, we will assume that each unary factor is specified by a **weighted finite-state automaton** (WFSA) whose weights fall in the semiring $(\mathbb{R}_{\geq 0}, +, \times)$. Thus the score $F_3(\ldots, V_{2SA} = x, \ldots)$ is the total weight of all paths in the F_3's WFSA that accept the string $x \in \Sigma_{2SA}^*$. Each path's weight is the product of its component arcs' weights, which are non-negative.

Similarly, we assume that each binary factor is specified by a **weighted finite-state transducer** (WFST). Such a model is essentially a generalization of stochastic edit distance (Ristad and Yianilos, 1996) in which the edit probabilities can be made sensitive to a finite summary of context.

Formally, a WFST is an automaton that resembles a weighted FSA, but it nondeterministically reads *two* strings x, y in parallel from left to right. The score of (x, y) is given by the total weight of all accepting paths in the WFST that map x to y. For example, different paths may consider various monotonic alignments of x with y, and we sum over these mutually exclusive possibilities.[1]

A factor might depend on $k > 2$ variables. This requires a k-tape **weighted finite-state machine** (WFSM), an obvious generalization where each path reads k strings in some alignment.[2]

To ensure that Z is finite in equation (1), we can require each factor to be a **"proper" WFSM**, i.e., its accepting paths have finite total weight (even if the WFSM is cyclic, with infinitely many paths).

[1] Each string is said to be on a different "tape," which has its own "read head," allowing the WFSM to maintain a separate position in each string. Thus, a path in a WFST may consume any number of characters from x before consuming the next character from y.

[2] Weighted acceptors and transducers are the cases $k = 1$ and $k = 2$, which are said to define **rational languages** and **rational relations**.

3.3 Parameters

Our probability model has trainable parameters: a vector of **feature weights** $\theta \in \mathbb{R}$. Each arc in each WFSM has a real-valued weight that depends on θ. Thus, tuning θ during training will change the arc weights, hence the path weights, the factor functions, and the whole probability distribution $p(\mathcal{A})$.

Designing the probability model includes specifying the topology and weights of each WFSM. Eisner (2002) explains how to specify and train such **parameterized WFSMs**. Typically, the weight of an arc is a simple sum like $\theta_{12} + \theta_{55} + \theta_{72}$, where θ_{12} is included on all arcs that share feature 12. However, more interesting parameterizations arise if the WFSM is constructed by operations such as transducer composition, or from a weighted regular expression.

3.4 Power of the formalism

Factored finite-state string models (1) were originally suggested by the second author, in Kempe et al. (2004). That paper showed that even in the unweighted case, such models could be used to encode relations that could not be recognized by any k-tape FSM. We offer a more linguistic example as a small puzzle. We invite the reader to specify a factored model (consisting of three FSTs as in Fig. 1) that assigns positive probability to just those triples of character strings (x, y, z) that have the form (*red_ball*, *ball_red*, *red*), (*white_house*, *house_white*, *white*), etc. This uses the auxiliary variable Z to help encode a relation between X and Y that swaps words of unbounded length. By contrast, no FSM can accomplish such unbounded swapping, even with 3 or more tapes.

Such extra power might be linguistically useful. Troublingly, however, Kempe et al. (2004) also observed that the framework is powerful enough to express computationally *undecidable* problems.[3] This implies that to work with arbitrary models, we will need approximate methods.[4] Fortunately, the graphical models community has already developed many such methods, to deal with the computational intractability (if not undecidability) of exact inference.

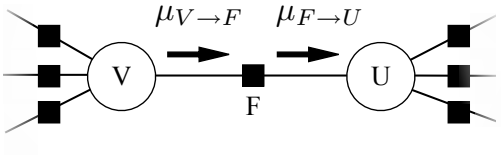


Figure 2: Illustration of messages being passed from variable to factor and factor to variable. Each message is represented by a finite-state acceptor.

4 Approximate Inference

In this paper, we focus on how **belief propagation** (BP)—a simple well-known method for approximate inference in MRFs (Bishop, 2006)—can be used in our setting. BP in its general form has not yet been widely used in the NLP community.[5] However, it is just a generalization to *arbitrary* factor graphs of the familiar forward-backward algorithm (which operates only on chain-structured factor graphs). The algorithm becomes approximate (and may not even converge) when the factor graphs have cycles. (In that case it is more properly called "loopy belief propagation.")

4.1 Belief propagation

We first sketch how BP works in general. Each variable V in the graphical model maintains a **belief** about its value, in the form of a marginal distribution $\tilde{p}_V$ over the possible values of V. The final beliefs are the output of the algorithm.

Beliefs arise from **messages** that are sent between the variables and factors along the edges of the factor graph. Variable V sends factor F a message $\mu_{V \to F}$, which is an (unnormalized) probability distribution over V's values v, computed by

$$\mu_{V \to F}(v) := \prod_{F' \in \mathcal{N}(V), F' \neq F} \mu_{F' \to V}(v) \quad (2)$$

where $\mathcal{N}$ is the set of neighbors of V in the graphical model. This message represents a consensus of V's *other* neighboring factors concerning V's value. It is how V tells F what its belief $\tilde{p}_V$ would be if F were absent. Informally, it communicates to F: *Here is what my value would be if it were up to my other neighboring factors F' to determine.*

[3] Consider a simple model with two variables and two binary factors: $p(V_1, V_2) \stackrel{\text{def}}{=} \frac{1}{Z} \cdot F_1(V_1, V_2) \cdot F_2(V_1, V_2)$. Suppose F_1 is 1 or 0 according to whether its arguments are equal. Under this model, $p(\epsilon) < 1$ iff there exists a string $x \neq \epsilon$ that can be transduced to itself by the unweighted transducer F_2. This question can be used to encode any instance of Post's Correspondence Problem, so is undecidable.

[4] Notice that the simplest approximation to cure undecidability would be to impose an arbitrary maximum on string length, so that the random variables have a finite domain, just as in most discrete graphical models.

[5] Notable exceptions are Sutton et al. (2004) for chunking and tagging, Sutton and McCallum (2004) for information extraction, Smith and Eisner (2008) for dependency parsing, and Cromierès and Kurohashi (2009) for alignment.

The factor F can then collect such incoming messages from neighboring variables and send its own message on to another neighbor U. Such a message $\mu_{F\to U}$ suggests good values for U, in the form of an (unnormalized) distribution over U's values u, computed by

$$\mu_{F\to U}(u) := \sum_{\mathcal{A}\ s.t. \mathcal{A}[U]=u} F(\mathcal{A}) \prod_{U'\in\mathcal{N}(F), U'\neq U} \mu_{U'\to F}(\mathcal{A}[U']) \tag{3}$$

where $\mathcal{A}$ is an assignment to all variables, and $\mathcal{A}[U]$ is the value of variable U in that assignment. This message represents F's prediction of U's value based on its *other* neighboring variables U'. Informally, via this message, F tells U: *Here is what I would like your value to be, based on the messages that my other neighboring variables have sent me about their values, and how I would prefer you to relate to them.*

Thus, each edge of the factor graph maintains two messages $\mu_{V\to F}, \mu_{F\to V}$. All messages are updated repeatedly, in some order, using the two equations above, until some stopping criterion is reached.[6] The *beliefs* are then computed:

$$\tilde{p}_V(v) \stackrel{\text{def}}{=} \prod_{F\in\mathcal{N}(V)} \mu_{F\to V}(v) \tag{4}$$

If variable V is *observed*, then the right-hand sides of equations (2) and (4) are modified to tell V that it *must* have the observed value v. This is done by multiplying in an extra message $\mu_{\text{obs}\to V}$ that puts probability 1 on v[7] and 0 on other values. That affects *other* messages and beliefs. The final belief at each variable estimates its *posterior* marginal under the MRF (1), *given all observations*.

4.2 Finite-state messages in BP

Both $\mu_{V\to F}$ and $\mu_{F\to V}$ are unnormalized distributions over the possible values of V—in our case, strings. A distribution over strings is naturally represented by a WFSA. Thus, belief propagation translates to our setting as follows:

- Each message is a WFSA.
- Messages are typically initialized to a one-state WFSA that accepts all strings in Σ^*, each with weight 1.[8]
- Taking a pointwise product of messages to V in equation (2) corresponds to WFSA intersection.
- If F in equation (3) is binary,[9] then there is only one U'. Then the outgoing message $\mu_{F\to U}$, a WFSA, is computed as $\text{domain}(F \circ \mu_{U'\to F})$.

  Here $\circ$ composes the factor WFST with the incoming message WFSA, yielding a WFST that gives a joint distribution over (U, U'). The domain operator projects this WFST onto the U side to obtain a WFSA, which corresponds to marginalizing to obtain a distribution over U.
- In general, F is a k-tape WFSM. Equation (3) "composes" $k-1$ of its tapes with $k-1$ incoming messages $\mu_{U'\to F}$, to construct a joint distribution over the k variables in $\mathcal{N}(F)$, then projects onto the k^{th} tape to marginalize over the $k-1$ U' variables and get a distribution over U. All this can be accomplished by the WFSM generalized composition operator $\boxtimes$ (Kempe et al., 2004).

After projecting, it is desirable to determinize the WFSA. Otherwise, the summation in (3) is only implicit—the summands remain as distinct paths in the WFSA[10]—and thus the WFSAs would get larger and larger as BP proceeds. Unfortunately, determinizing a WFSA still does not guarantee a small result. In fact it can lead to exponential blowup, or even infinite blowup.[11] Thus, in practice we recommend against determinizing the messages, which may be inherently complex. To shrink a message, it is safer to *approximate* it with a small deterministic WFSA, as discussed in the next section.

4.3 Approximation of messages

In our domain, it is possible for the finite-state messages to *grow unboundedly in size* as they flow around a cycle. After all, our messages are not just multinomial distributions over a fixed finite

[6] Preferably when the beliefs converge to some fixed point (a local minimum of the Bethe free energy). However, convergence is not guaranteed.

[7] More generally, on *all* possible observed variables.

[8] This is an (improper) uniform distribution over Σ^*. Although is *not* a proper WFSA (see section 3.2), there is an upper bound on the weights it assigns to strings. That guarantees that all the messages and beliefs computed by (2)–(4) will be proper FSMs, provided that all the factors are proper WFSMs.

[9] If it is unary, (3) trivially reduces to $\mu_{F\to U} = F$.

[10] The usual implementation of projection does not change the topology of the WFST, but only deletes the U' part of its arc labels. Thus, multiple paths that accept the same value of U remain distinct according to the distinct values of U' that they were paired with before projection.

[11] If there is no deterministic equivalent (Mohri, 1997).

set. They are distributions over the infinite set Σ^*. A WFSA represents this in finite space, but more complex distributions require bigger WFSAs, with more distinct states and arc weights.

Facing the same problem for distributions over the infinite set $\mathbb{R}$, Sudderth et al. (2002) simplified each message $\mu_{V \to F}$, approximating a complex Gaussian mixture by using fewer components.

We could act similarly, variationally approximating a large WFSA P with a smaller one Q. Choose a family of message approximations (such as bigram models) by specifying the topology for a (small) deterministic WFSA Q. Then choose Q's edge weights to minimize the KL divergence $\mathrm{KL}(P \,\|\, Q)$. This can be done in closed form.[12]

Another possible procedure—used in the experiments of this paper—approximates $\mu_{V \to F}$ by pruning it back to a finite set of most plausible strings.[13] Equation (2) requests an intersection of several WFSAs, e.g., $\mu_{F_1 \to V} \cap \mu_{F_2 \to V} \cap \cdots$. List all strings that appear on any of the 1000-best paths in any of these WFSAs, removing duplicates. Let $\bar{Q}$ be a uniform distribution over this combined list of plausible strings, represented as a determinized, minimized, acyclic WFSA. Now approximate the intersection of equation (2) as $((\bar{Q} \cap \mu_{F_1 \to V}) \cap \mu_{F_2 \to V}) \cap \cdots$. This is efficient to compute and has the same topology as $\bar{Q}$.

5 Training the Model Parameters

Any standard training method for MRFs will transfer naturally to our setting. In all cases we draw on Eisner (2002), who showed how to train the parameters θ of a *single* WFST, F, to (locally) maximize the joint or conditional probability of fully or partially observed training data. This involves computing the gradient of that likelihood function with respect to θ.[14]

We must generalize this to train a *product* of WFSMs. Typically, training data for an MRF (1) consists of some fully or partially observed IID samples of the joint distribution $p(V_1, \ldots V_n)$. It is well-known how to tune an MRF's parameters θ by stochastic gradient descent to locally maximize the probability of this training set, even though both the probability and its gradient are in general intractable to compute in an MRF. The gradient is a sum of quantities, one for each factor F_j. While the summand for F_j cannot be computed exactly, it can be estimated using the BP messages to F_j. Roughly speaking, the gradient for F_j is computed much as in supervised training (see above), but treating any message $\mu_{V_i \to F_j}$ as an uncertain observation of V_i—a form of noisy supervision.[15]

Our concerns about training are the same as for any MRF. First of all, BP is approximate. Kulesza and Pereira (2008) warn that its estimates of the gradient can be misleading. Second, semi-supervised training (which we will attempt below) is always difficult and prone to local optima. As in EM, a small number of supervised examples for some variable may be drowned out by many noisily reconstructed examples.

Faster and potentially more stable approaches include the **piecewise training** methods of Sutton and McCallum (2008), which train the factors independently or in small groups. In the semi-supervised case, each factor can be trained on only the supervised forms available for it. It might be useful to reweight the trained factors (cf. Smith et al. (2005)), or train the factors consecutively (cf. Fahlman and Lebiere (1990)), in a way that minimizes the loss of BP decoding on held-out data.

6 Comparison With Other Approaches

6.1 Multi-tape WFSMs

In principle, one could use a 100-tape WFSM to jointly model the 100 distinct forms of a typical Polish verb. In other words, the WFSM would describe the distribution of a random variable $\vec{V} = \langle V_1, \ldots, V_{100} \rangle$, where each V_i is a string. One would train the parameters of the WFSM on a sample of $\vec{V}$, each sample being a fully or partially observed paradigm for some Polish verb. The resulting distribution could be used to infer missing forms for these or other verbs.

[12] See Li et al. (2009, footnote 9) for a sketch of the construction, which finds locally normalized edge weights. Or if Q is large but parameterized by some compact parameter vector ϕ, so we are only allowed to control its edge weights via ϕ, then Li and Eisner (2009, section 6) explain how to minimize $\mathrm{KL}(P \,\|\, Q)$ by gradient descent. In both cases Q must be deterministic.

We remark that if a factor F were specified by a synchronous grammar rather than a WFSM, then its outgoing messages would be weighted context-free languages. Exact intersection of these is undecidable, but they too can be approximated variationally by WFSAs, with the same methods.

[13] We are also considering other ways of adaptively choosing the topology of WFSA approximations at runtime, particularly in conjunction with expectation propagation.

[14] The likelihood is usually non-convex; even when the two strings are observed (supervised training), their accepting path through the WFST may be ambiguous and unobserved.

[15] See Bishop (2006), or consult Smith and Eisner (2008) for notation close to that of this paper.

As a simple example, either a morphological generator or a morphological analyzer might need the probability that *krzyczałoby* is the neuter third-person singular conditional imperfective of *krzyczeć*, despite never having observed it in training. The model determines this probability based on other observed and hypothesized forms of *krzyczeć*, using its knowledge of how neuter third-person singular conditional imperfectives are related to these other forms in other verbs.

Unfortunately, such a 100-tape WFSM would be huge, with an astronomical number of arcs (each representing a possible 100-way edit operation). Our approach is to factor the problem into a number of (e.g.) pairwise relationships among the verb forms. Using a factored distribution has several benefits over the k-tape WFSM: (1) a smaller representation in memory, (2) a small number of parameters to learn, (3) efficient approximate computation that takes advantage of the factored structure, (4) the ability to reuse WFSAs and WFSTs previously developed for smaller problems, (5) additional modeling power.

6.2 Simpler graphical models on strings

Some previous researchers have used factored joint models of several strings. To our knowledge, they have all chosen *acyclic*, *directed* graphical models. The acyclicity meant that exact inference was at least possible for them, if not necessarily efficient. The factors in these past models have been WFSTs (though typically simpler than the ones we will use).

Many papers have used cascades of probabilistic finite-state transducers. Such a cascade may be regarded as a directed graphical model with a linear-chain structure. Pereira and Riley (1997) built a speech recognizer in this way, relating acoustic to phonetic to lexical strings. Similarly, Knight and Graehl (1997) presented a generative cascade using 4 variables and 5 factors: $p(w, e, j, k, o) \stackrel{\text{def}}{=} p(w) \cdot p(e \mid w) \cdot p(j \mid e) \cdot p(k \mid j) \cdot p(o \mid k)$ where e is an English word sequence, w its pronunciation, j a Japanese version of the pronunciation, k a katakana rendering of the Japanese pronunciation, and o an OCR-corrupted version of the katakana. Knight and Graehl used finite-state operations to perform inference at test time, observing o and recovering the most likely w, while marginalizing out e, j, and k.

Bouchard-Côté et al. (2009) reconstructed ancient word forms given modern equivalents. They used a directed graphical model, whose tree structure reflected the evolutionary development of the modern languages, and which included latent variables for historical intermediate forms that were never observed in training data. They used Gibbs sampling rather than an exact solution (possible on trees) or a variational approximation (like our BP).

Our work seeks to be general in terms of the graphical model structures used, as well as efficient through the use of BP with approximate messages. We also seek to avoid local normalization, using a globally normalized model.[16]

6.3 Unbounded objects in graphical models

We distinguish our work from "dynamic" graphical models such as Dynamic Bayesian Networks and Conditional Random Fields, where the string *brechen* would be represented by creating 7 letter-valued variables. Those methods can represent strings (or paths) of any length—but the length for each training or test string must be specified in advance, not inferred. Furthermore, it is awkward and costly to model unknown alignments, since the variables are position-specific, and any position in *brechen* could in principle align with any position in *brichst*. WFSTs are a much more natural and flexible model of string pairs.

We also distinguish our work from current nonparametric Bayesian models, which sometimes generate unbounded strings, trees, or grammars. If they generate two unbounded objects, they model their relationship by a single synchronous generation process (akin to Section 6.1), rather than by a globally normalized product of overlapping factors.

7 Experiments

To study our approach, we conducted initial experiments that reconstruct missing word forms in morphological paradigms. In *inflectional morphology*, each uninflected verb form (**lemma**) is associated with a vector of forms that are inflected for tense, person, number, etc. Some inflected forms may be observed frequently in natural text, others rarely. Two variables that are usually predictable from each other may or may not keep this relationship in the case of an irregular verb.

[16] Although we do normalize locally during piecewise training (see section 7.3).

(a) # paradigms	9,393
(b) # finite forms per paradigm	9
(c) # hidden finite forms per paradigm (avg.)	8.3
(d) # paradigms with some finite form(s) observed	2,176
(e) In (d), # of finite forms observed (avg.)	3.4

Table 1: Statistics of our training data.

Our task is to reconstruct (*generate*) specific unobserved morphological forms in a paradigm by learning from observed ones. This is a particularly interesting semisupervised scenario, because different subsets of the variables are observed on different examples.

7.1 Experimental data

We used orthographic rather than phonological forms. We extracted morphological paradigms for all 9393 German verbs in the CELEX morphological database. Each paradigm lists 5 present-tense and 4 past-tense indicative forms, as well as the verb's lemma, for a total of 10 string-valued variables.[17] In each paradigm, we removed, or hid, verb forms that occur only rarely in natural text, i.e, verb forms with a small frequency figure provided by CELEX.[18] All paradigms other than *sein* ('to be') were now incompletely observed. Table 1 gives some statistics.

7.2 Model factors and parameters

Our current MRF uses only binary factors. Each factor is a WFST that is trained to relate 2 of the 10 variables (morphological forms). Each WFST can score an aligned pair using a log-linear model that counts features in a sliding 3-character window. To score an unaligned pair, it sums over all possible alignments. Specifically, our WFST topology and parameterization follow the state-of-the-art approach to *supervised* morphology in Dreyer et al. (2008), although we dropped some of their features to speed up these early experiments.[19] We implemented and manipulated all WFSMs using the OpenFST library (Allauzen et al., 2007).

7.3 Training in the experiments

We trained θ on the incompletely observed paradigms. As suggested in section 5, we used a variant of piecewise pseudolikelihood training (Sutton and McCallum, 2008). Suppose there is a binary factor F attached to forms U and V. For any value of θ, we can define $p_{UV}(U \mid V)$ from the tiny MRF consisting only of U, V, and F. We can therefore compute the goodness $L_{UV} \stackrel{\text{def}}{=} \log p_{UV}(u_i \mid v_i) + \log_{VU}(v_i \mid u_i)$,[20] summed over all *observed* (U, V) pairs in training data. We attempted to tune θ to maximize the total L_{UV} over all U, V pairs,[21] regularized by subtracting $||\theta||^2$. Note that different factors thus enjoyed different amounts of observed training data, but training was fully supervised (except for the unobserved alignments between u_i and v_i).

7.4 Inference in the experiments

At test time, we are given each lemma (e.g. *brechen*) and all its observed (frequent) inflected forms (e.g., *brachen*, *bricht*,...), and are asked to predict the remaining (rarer) forms (e.g., *breche, brichst,* ...).

We run approximate joint inference using belief propagation.[22] We extract our output from the final beliefs: for each unseen variable V, we pre-

[17] Some pairs of forms are always identical in German, hence are treated as a single form by CELEX. We likewise use a single variable—these are the "1,3" variables in Fig. 3.

Occasionally a form is listed as UNKNOWN. We neither train nor evaluate on such forms, although the model will still predict them.

[18] The frequency figure for each word form is based on counts in the Mannheim News corpus. We hide forms with frequency < 10.

[19] We dropped their latent classes and regions as well as features that detected which characters were orthographic vowels. Also, we retained their "target language model features" only in the baseline "U" model, since elsewhere they seemed to hurt in our current training setup.

We followed Dreyer et al. (2008) in slightly pruning the space of possible alignments. We compensated by replacing their WFST, F, with the union $F \cup 10^{-12}(0.999\Sigma \times \Sigma)^*$. This ensured that the factor could still map any string to any other string (though perhaps with very low weight), guaranteeing that the intersection at the end of section 4.3 would be non-empty.

[20] The second term is omitted if V is the lemma. We do not train the model to predict the lemma since it is always observed in test data.

[21] Unfortunately, just before press time we discovered that this was not quite what we had done. A shortcut in our implementation trained $p_{UV}(U \mid V)$ and $p_{VU}(V \mid U)$ separately. This let them make different use of the (unobserved) alignments—so that even if each individually liked the pair (u, v), they might not have been able to agree on the same accepting path for it at test time. This could have slightly harmed our joint inference results, though not our baselines.

[22] To derive the update order for message passing, we take an arbitrary spanning tree over the factor graph, and let O be a list of all factors and variables that is topologically sorted according to the spanning tree, with the leaves of the tree coming first. We then discard the spanning tree. A single iteration visits all factors and variables in order of O, updating each one's messages to later variables and factors, and then visits all factors and variables in reverse order, updating each one's messages to earlier variables and factors.

dict its value to be $\text{argmax}_v \tilde{p}_V(v)$. This prediction considers the values of all other unseen variables but sums over their possibilities. This is the Bayes-optimal decoder for our scoring function, since that function reports the fraction of *individual forms* that were predicted *perfectly*.[23]

7.5 Model selection of MRF topology

It is hard to know *a priori* what the causal relationships might be in a morphological paradigm. In principle, one would like to *automatically* choose which factors to have in the MRF. Or one could start with many factors, but use methods such as those suggested in section 5 to learn that certain less useful factors should be left weak to avoid confusing loopy BP.

For our present experiments, we simply compared several fixed model topologies (Fig. 3). These were variously unconnected (U), chain graphs (C1,. . . , C4), trees (T1, T2), or loopy graphs (L1,. . . , L4). We used several factor graphs that differ only by one or two added factors and compared the results. The graphs were designed by hand; they connect some forms with similar morphological properties more or less densely.

We trained different models using the observed forms in the 9393 paradigms as training data. The first 100 paradigms were then used as development data for *model selection*:[24] we were given the answers to their hidden forms, enabling us to compare the models. The best model was then evaluated on the 9293 remaining paradigms.

7.6 Development data results

The models are compared on development data in Table 2. Among the factor graphs we evaluated, we find that L4 (see Fig. 3) performs best overall (whole-word accuracy 82.1). Note that the unconnected graph U does not perform very well (69.0), but using factor graphs with more connecting factors generally helps overall accuracy (see C1–C3). Note, however, that in some cases the additional structure hurts: The chain model C4 and the loopy model L1 perform relatively badly. The

[23] If we instead wished to maximize the fraction of entire paradigms that were predicted perfectly, then we would have approximated full MAP decoding over the paradigm (Viterbi decoding) by using max-product BP. Other loss functions (e.g., edit distance) would motivate other decoding methods.

[24] Using these paradigms was simply a quick way to avoid model selection by cross-validation. If data were really as sparse as our training setup pretends (see Table 2), then 100 complete paradigms would be too valuable to squander as mere development data.

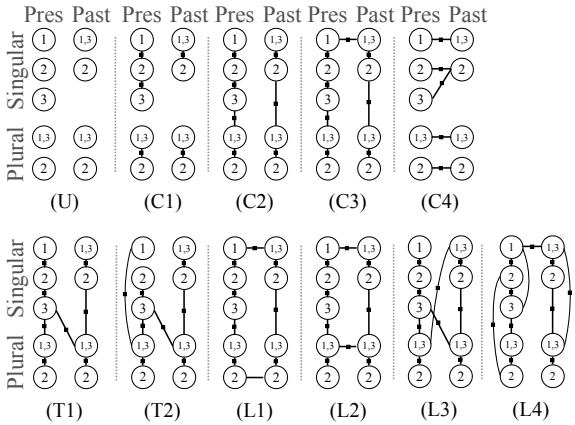


Figure 3: The graphs that we evaluate on development data. The nodes represent morphological forms, e.g. the first node in the left of each graph represents the first person singular present. Each variable is also connected to the lemma (not shown). See results in Table 2.

reason for such a performance degradation is that undertrained factors were used: The factors relating second-person to second-person forms, for example, are trained from only 8 available examples.

Non-loopy models always converge (exactly) in one iteration (see footnote 22). But even our loopy models appeared to converge in accuracy within two iterations. Only L3 and L4 required the second iteration, which made tiny improvements.

7.7 Test data results

Based on the development results, we selected model L4 and tested on the remaining 9293 paradigms.

We regard the unconnected model U as a baseline to improve upon. We also tried a rather different baseline as in (Dreyer et al., 2008). We trained the machine translation toolkit Moses (Koehn et al., 2007) to translate groups of letters rather than groups of words ("phrases"). For each form f to be predicted, we trained a Moses model on all supervised form pairs (l, f) available in the data, to learn a prediction for the form given the lemma l. The M,3 condition restricted Moses use "phrases" no longer than 3 letters, comparable to our own trigram-based factors (see section 7.2). M,15 could use up to 15 letters.

Again, our novel L4 model far outperformed the others overall. Breaking the results down by form, we find that this advantage mainly comes from the 3 forms with the fewest observed training examples (Table 3, first 3 rows). The M and U models are barely able to predict these forms at all from the lemma, but L4 can predict them bet-

Unconn.	Chains				Trees		Loops			
U	C1	C2	C3	C4	T1	T2	L1	L2	L3	L4
69.0	72.9	73.4	74.8	65.2	78.1	78.7	62.3	79.6	78.9	**82.1**

Table 2: Whole-word accuracies of the different models in reconstructing the missing forms in morphological paradigms, here on 100 verbs (development data). The names refer to the graphs in Fig. 3. We selected L4 as final model (Table 3).

Form	# obs.	M,3	M,15	U	L4
2.Sg.Pa.	4	0.0	0.2	0.8	**69.7**
2.Pl.Pa.	9	0.9	1.1	1.4	**45.6**
2.Sg.Pr.	166	49.4	62.6	74.7	**90.5**
1.Sg.Pr.	285	**99.6**	98.8	99.3	97.2
1,3.Pl.Pa.	673	46.5	**78.3**	75.0	75.6
1,3.Sg.Pa.	1124	65.0	**88.8**	84.0	74.8
2.Pl.Pr.	1274	98.3	**99.2**	99.0	96.4
3.Sg.Pr.	1410	91.0	**95.9**	95.2	88.2
1,3.Pl.Pr.	1688	**99.8**	98.9	**99.8**	98.0
All	6633	59.2	67.3	68.0	**81.2**

Table 3: Whole-word accuracies on the missing forms from 9293 test paradigms. The Moses baselines and our unconnected model (U) predict each form separately from the lemma, which is always observed. L4 uses all observations jointly, running belief propagation for decoding. Moses,15 memorizes phrases of length up to 15, all other models use max length 3. The table is sorted by the column "# obs.", which reports the numbers of observations for a given form.

ter by exploiting other observed or latent forms. By contrast, well-trained forms were already easy enough for the M and U models that L4 had little new to offer and in fact suffered from its approximate training and/or inference.

Leaving aside the comparisons, it was useful to confirm that loopy BP could be used in this setting at all. 8014 of the 9293 test paradigms had ≤ 2 observed forms (in addition to the lemma) but ≥ 7 missing forms. One might have expected that loopy BP would have failed to converge, or converged to the wrong thing. Nonetheless, it achieved quite respectable success at exactly predicting various inflected forms.

For the curious, Table 4 shows accuracies grouped by different categories of paradigms, where the category is determined by the number of missing forms to predict. Most paradigms fall in the category where 7 to 9 forms are missing, so the accuracies in that line are similar to the overall accuracies in Table 3.

# missing	# paradig.	M,3	M,15	U	L4
1–3	205	20.3	20.8	26.8	**74.4**
4–6	1037	44.2	50.5	52.7	**82.8**
7–9	8014	60.6	68.8	69.4	**81.1**

Table 4: Accuracy on test data, reported separately for paradigms in which 1–3, 4–6, or 7–9 forms are missing. Missing words have CELEX frequency count < 10; these are the ones to predict. (The numbers in col. 2 add up to 9256, not 9293, since some paradigms are incomplete in CELEX to begin with, with no forms to be removed or evaluated.)

8 Conclusions

We have proposed that one can jointly model several multiple strings by using Markov Random Fields. We described this formally as an undirected graphical model with string-valued variables and whose factors (potential functions) are defined by weighted finite-state transducers. Each factor evaluates some subset of the strings.

Approximate inference can be done by loopy belief propagation. The messages take the form of weighted finite-state acceptors, and are constructed by standard operations. We explained why the messages might become large, and gave methods for approximating them with smaller messages. We also discussed training methods.

We presented some pilot experiments on the task of jointly predicting multiple missing verb forms in morphological paradigms. The factors were simplified versions of statistical finite-state models for supervised morphology. Our MRF for this task might be used not only to conjugate verbs (e.g., in MT), but to guide further learning of morphology—either active learning from a human or semi-supervised learning from the distributional properties of a raw text corpus.

Our modeling approach is potentially applicable to a wide range of other tasks, including transliteration, phonology, cognate modeling, multiple-sequence alignment and system combination.

Our work ties into a broader vision of using algorithms like belief propagation to coordinate the work of several NLP models and algorithms. Each individual factor considers some portion of a joint problem, using classical statistical NLP methods (weighted grammars, transducers, dynamic programming). The factors coordinate their work by passing marginal probabilities. Smith and Eisner (2008) reported complementary work in this vein.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proc. of CIAA*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Alexandre Bouchard-Côté, Thomas L. Griffiths, and Dan Klein. 2009. Improved reconstruction of protolanguage word forms. In *Proc. of HLT-NAACL*, pages 65–73, Boulder, Colorado, June. Association for Computational Linguistics.

Fabien Cromierès and Sadao Kurohashi. 2009. An alignment algorithm using belief propagation and a structure-based distortion model. In *Proc. of EACL*, pages 166–174, Athens, Greece, March. Association for Computational Linguistics.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proc. of EMNLP*, Honolulu, Hawaii, October.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proc. of ACL*, pages 1–8, Philadelphia, July.

Scott E. Fahlman and Christian Lebiere. 1990. The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University.

André Kempe, Jean-Marc Champarnaud, and Jason Eisner. 2004. A note on join and auto-intersection of n-ary rational relations. In Loek Cleophas and Bruce Watson, editors, *Proceedings of the Eindhoven FASTAR Days (Computer Science Technical Report 04-40)*. Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Netherlands.

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In *Proc. of ACL*, pages 128–135.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL, Companion Volume*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Alex Kulesza and Fernando Pereira. 2008. Structured learning with approximate inference. In *Proc. of NIPS*.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proc. of EMNLP*.

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009. Variational decoding for statistical machine translation. In *Proc. of ACL*.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2).

Fernando C. N. Pereira and Michael Riley. 1997. Speech recognition by composition of weighted finite automata. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*. MIT Press, Cambridge, MA.

Eric Sven Ristad and Peter N. Yianilos. 1996. Learning string edit distance. Technical Report CS-TR-532-96, Princeton University, Department of Computer Science, October.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *Proc. of ACL*, pages 18–25, June.

Erik B. Sudderth, Alexander T. Ihler, Er T. Ihler, William T. Freeman, and Alan S. Willsky. 2002. Nonparametric belief propagation. In *Proc. of CVPR*, pages 605–612.

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*.

Charles Sutton and Andrew McCallum. 2008. Piecewise training for structured prediction. *Machine Learning*. In submission.

Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proc. of ICML*.

Reverse Engineering of Tree Kernel Feature Spaces

Daniele Pighin
FBK-Irst, HLT
Via di Sommarive, 18 I-38100 Povo (TN) Italy
pighin@fbk.eu

Alessandro Moschitti
University of Trento, DISI
Via di Sommarive, 14 I-38100 Povo (TN) Italy
moschitti@disi.unitn.it

Abstract

We present a framework to extract the most important features (tree fragments) from a Tree Kernel (TK) space according to their importance in the target kernel-based machine, e.g. Support Vector Machines (SVMs). In particular, our mining algorithm selects the most relevant features based on SVM estimated weights and uses this information to automatically infer an explicit representation of the input data. The explicit features (a) improve our knowledge on the target problem domain and (b) make large-scale learning practical, improving training and test time, while yielding accuracy in line with traditional TK classifiers. Experiments on semantic role labeling and question classification illustrate the above claims.

1 Introduction

The last decade has seen a massive use of Support Vector Machines (SVMs) for carrying out NLP tasks. Indeed, their appealing properties such as 1) solid theoretical foundations, 2) robustness to irrelevant features and 3) outperforming accuracy have been exploited to design state-of-the-art language applications.

More recently, kernel functions, which implicitly represent data in some high dimensional space, have been employed to study and further improve many natural language systems, e.g. (Collins and Duffy, 2002), (Kudo and Matsumoto, 2003), (Cumby and Roth, 2003), (Cancedda et al., 2003), (Culotta and Sorensen, 2004), (Toutanova et al., 2004), (Kazama and Torisawa, 2005), (Shen et al., 2003), (Gliozzo et al., 2005), (Kudo et al., 2005), (Moschitti et al., 2008), (Diab et al., 2008). Unfortunately, the benefit to easily and effectively model the target linguistic phenomena is reduced by the the implicit nature of the kernel space, which prevents to directly observe the most relevant features. As a consequence, even very accurate models generally fail in providing useful feedback for improving our understanding of the problems at study. Moreover, the computational burden induced by high dimensional kernels makes the application of SVMs to large corpora still more problematic.

In (Pighin and Moschitti, 2009), we proposed a feature extraction algorithm for Tree Kernel (TK) spaces, which selects the most relevant features (tree fragments) according to the gradient components (weight vector) of the hyperplane learnt by an SVM, in line with current research, e.g. (Rakotomamonjy, 2003; Weston et al., 2003; Kudo and Matsumoto, 2003). In particular, we provided algorithmic solutions to deal with the huge dimensionality and, consequently, high computational complexity of the fragment space. Our experimental results showed that our approach reduces learning and classification processing time leaving the accuracy unchanged.

In this paper, we present a new version of such algorithm which, under the same parameterization, is almost three times as fast while producing the same results. Most importantly, we explored tree fragment spaces for two interesting natural language tasks: Semantic Role Labeling (SRL) and Question Classification (QC). The results show that: (a) on large data sets, our approach can improve training and test time while yielding almost unaffected classification accuracy, and (b) our framework can effectively exploit the ability of TKs and SVMs to, respectively, generate and recognize relevant structured features. In particular, we (i) study in more detail the relevant fragments identfied for the boundary classification task of SRL, (ii) closely observe the most relevant fragments for each QC class and (iii) look at the diverse syntactic patterns characterizing each ques-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 111–120,
Singapore, 6-7 August 2009.

tion category.

The rest of the paper is structured as follows: Section 2 will briefly review SVMs and TK functions; Section 3 will detail our proposal for the linearization of a TK feature space; Section 4 will review previous work on related subjects; Section 5 will detail the outcome of our experiments, and Section 6 will discuss some relevant aspects of the evaluation; finally, in Section 7 we will draw our conclusions.

2 Tree Kernel Functions

The decision function of an SVM is:

$$f(\vec{x}) = \vec{w} \cdot \vec{x} + b = \sum_{i=1}^{n} \alpha_i y_i \vec{x_i} \cdot \vec{x} + b \qquad (1)$$

where $\vec{x}$ is a classifying example and $\vec{w}$ and b are the separating hyperplane's *gradient* and its *bias*, respectively. The gradient is a linear combination of the training points $\vec{x_i}$, their labels y_i and their weights α_i. Applying the so-called *kernel trick* it is possible to replace the scalar product with a *kernel function* defined over pairs of *objects*:

$$f(o) = \sum_{i=1}^{n} \alpha_i y_i \mathrm{k}(o_i, o) + b$$

with the advantage that we do not need to provide an explicit mapping $\phi(\cdot)$ of our examples in a vector space.

A Tree Kernel function is a convolution kernel (Haussler, 1999) defined over pairs of trees. Practically speaking, the kernel between two trees evaluates the number of substructures (or *fragments*) they have in common, i.e. it is a measure of their overlap. The function can be computed recursively in closed form, and quite efficient implementations are available (Moschitti, 2006). Different TK functions are characterized by alternative fragment definitions, e.g. (Collins and Duffy, 2002) and (Kashima and Koyanagi, 2002). In the context of this paper we will be focusing on the SubSet Tree (SST) kernel described in (Collins and Duffy, 2002), which relies on a fragment definition that does not allow to break production rules (i.e. if any child of a node is included in a fragment, then also all the other children have to). As such, it is especially indicated for tasks involving constituency parsed texts.

Implicitly, a TK function establishes a correspondence between distinct fragments and dimensions in some *fragment space*, i.e. the space of all the possible fragments. To simplify, a tree t can be represented as a vector whose attributes count the occurrences of each fragment within the tree. The kernel between two trees is then equivalent to the scalar product between pairs of such vectors, as exemplified in Figure 1.

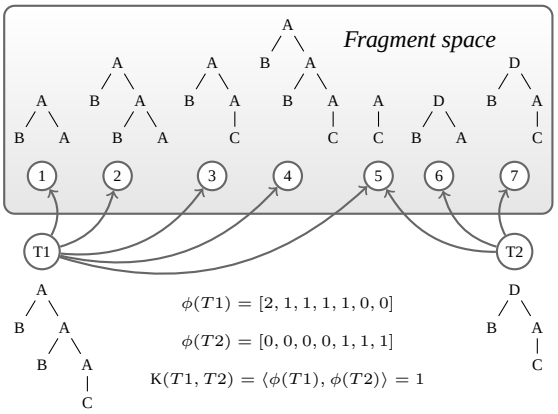


Figure 1: Esemplification of a fragment space and the kernel product between two trees.

3 Linearization of a TK function

Our objective is to efficiently mine the most relevant fragments from the huge fragment space, so that we can explicitly represent our input trees in terms of these fragments and learn fast and accurate linear classifiers.

The framework defines five distinct activities, detailed in the following paragraphs.

3.1 Kernel Space Learning (*KSL*)

The first step involves the generation of an approximation of the whole fragment space, i.e. we can consider only the trees that encode the most relevant fragments. To this end, we can partition our training data into S smaller sets, and use the SVM and the SST kernel to learn S models. We will only consider the fragments encoded by the support vectors of the S models. In the next stage, we will use the SVM estimated weights to drive our feature selection process.

Since time complexity of SVM training is approximately quadratic in the number of examples, by breaking training data into smaller sets we can considerably accelerate the process of filtering trees and estimating support vector weights. According to statistical learning theory, being trained on smaller subsets of the available data these models will be less robust with respect to the minimization of the empirical risk (Vapnik, 1998).

Algorithm 3.1: MINE_MODEL(M, L, λ)

```
global maxexp
prev ← ∅ ; CLEAR_INDEX()
for each ⟨αy, t⟩ ∈ M
do { T_i ← α · y/‖t‖
     for each n ∈ N_t
     do { f ← FRAG(n) ; rel = λ · T_i
          prev ← prev ∪ {f, rel}
          PUT(f, rel) }
best_pr ← BEST(L) ;
while true
do { next ← ∅
     for each ⟨f, rel⟩ ∈ prev if f ∈ best_pr
     do { X = EXPAND(f, maxexp)
          rel_exp ← λ · rel
          for each frag ∈ X
          do { temp = {frag, rel_exp}
               next ← next ∪ temp
               PUT(frag, rel_exp) }
     best ← BEST(L)
     if not CHANGED()
       then break
     best_pr ← best
     prev ← next }
F_L ← best_pr
return (F_L)
```

Nonetheless, since we do not need to employ them for classification (but just to direct our feature selection process, as we will describe shortly), we can accept to rely on sub-optimal weights. Furthermore, research results in the field of SVM parallelization using cascades of SVMs (Graf et al., 2004) suggest that support vectors collected from locally learnt models can encode many of the relevant features retained by models learnt globally. Henceforth, let M_s be the model associated with the s-th split, and $\mathcal{F}_s$ the fragment space that can describe all the trees in M_s.

3.2 Fragment Mining and Indexing (*FMI*)

In Equation 1 it is possible to isolate the gradient $\vec{w} = \sum_{i=1}^{n} \alpha_i y_i \vec{x_i}$, with $\vec{x_i} = [x_i^{(1)}, \ldots, x_i^{(N)}]$, N being the dimensionality of the feature space. For a tree kernel function, we can rewrite $x_i^{(j)}$ as:

$$x_i^{(j)} = \frac{t_{i,j}\lambda^{\ell(f_j)}}{\|t_i\|} = \frac{t_{i,j}\lambda^{\ell(f_j)}}{\sqrt{\sum_{k=1}^{N}(t_{i,k}\lambda^{\ell(f_k)})^2}} \quad (2)$$

where: $t_{i,j}$ is the number of occurrences of the fragment f_j, associated with the j-th dimension of the feature space, in the tree t_i; λ is the kernel decay factor; and $\ell(f_j)$ is the depth of the fragment.

The relevance $|w^{(j)}|$ of the fragment f_j can be measured as:

$$|w^{(j)}| = \left|\sum_{i=1}^{n} \alpha_i y_i x_i^{(j)}\right| = \frac{\left|\sum_{i=1}^{n} \alpha_i y_i t_{i,j} \lambda^{\ell(f_j)}\right|}{\|t_i\|} . \quad (3)$$

We fix a threshold L and from each model M_s (learnt during KSL) we select the L most relevant fragments, i.e. we build the set $\mathcal{F}_{s,L} = \cup_k \{f_k\}$ so that:

$$|\mathcal{F}_{s,L}| = L \text{ and } |w^{(k)}| \geq |w^{(i)}| \forall f_i \in \mathcal{F} \setminus \mathcal{F}_{s,L} .$$

To generate all the fragments encoded in a model, we adopt the greedy strategy described in Algorithm 3.1. Its arguments are: an SVM model M represented as $\langle \alpha y, t \rangle$ pairs, where t is a tree structure; the threshold value L; and the kernel decay factor λ.

The function FRAG(n) generates the smallest fragment rooted in node n (i.e. for an SST kernel, the fragment consisting of n and its direct children). We call such fragment a *base* fragment. The function EXPAND($f, maxexp$) generates all the fragments that can be derived from the fragment f by expanding, i.e. including in the fragment the direct children of some of its nodes. These fragments are *derived* from f. The parameter $maxexp$ limits fragment proliferation by setting the maximum number of nodes which can be expanded in a fragment expansion operation. For example, if there are 10 nodes which can be expanded in fragment f, then only the fragments where at most 3 of the 10 nodes are expanded will be generated by a call to EXPAND($f, 3$).

Every time we generate a fragment f, the function PUT(f, rel) saves the fragment along with its relevance rel in an *index*. The index keeps track of the cumulative relevance of a fragment, and its implementation has been optimized for fast insertions and spatial compactness.

A whole cycle of expansions is considered as an iteration of the mining process: we take into account all the fragments that have undergone k expansions and produce all the fragments that result from a further expansion, i.e. all the fragments expanded $k+1$ times.

We keep iterating until we reach a stop criterion, which we base on the threshold value L, i.e. the limit on the number of fragments that we are interested in mining from a model. During each iteration $k+1$, we only expand the best L fragments identified during the previous iteration k. When

the iteration is complete we re-evaluate the set of L best fragments in the index, and we stop only if the worst of them, i.e. the L-th ranked fragment at the step $k+1$, and its score are the same as at the end of the previous iteration. That is, we assume that if none of the fragments mined during the $(k+1)$-th iteration managed to affect the bottom of the pool of the L most relevant fragments, then none of their expansions is likely to succeed. In the algorithm, $\mathcal{N}_t$ is the set of nodes of the tree t; BEST(L) returns the L highest ranked fragments in the index; CHANGED$()$ verifies whether the bottom of the L-best set has been affected by the last iteration or not.

We call MINE_MODEL$(\cdot)$ on each of the models M_s that we learnt from the S initial splits. For each model, the function returns the set of L-best fragments in the model. The union of all the fragments harvested from each model is then saved into a dictionary $\mathcal{D}_L$ which will be used by the next stage.

3.2.1 Discussion on FMI algorithm

With respect to the algorithm presented in (Pighin and Moschitti, 2009), the one presented here has the following advantages:

- the process of building fragments is strictly small-to-large: fragments that span $n+1$ levels of the tree may be generated only after all those spanning n levels;
- the threshold value L is a parameter of the mining process, and it is used to prevent the algorithm from generating more fragments than necessary, thus making it more efficient;
- it has one less parameter ($maxdepth$) which was used to force fragments to span at-most a given number of levels. The new algorithm does not need it since the maximum number of iterations is implicitly set via L.

These differences result in improved efficiency for the FMI stage. For example, on the data for the boundary classification task (see Section 5), using comparable parameters the old algorithm required 85 minutes to mine the most relevant fragments, whereas the new one only takes 31, i.e. it is 2.74 times as fast.

3.3 Tree Fragment Extraction (*TFX*)

During this phase we actually linearize our data: a file encoding label-tree pairs $\langle y_i, t_i \rangle$ is transformed to encode label-vector pairs $\langle y_i, \vec{v_i} \rangle$. To do so, we generate the fragment space of t_i, using a variant of the mining algorithm described in Algorithm 3.1, and encode in $\vec{v_i}$ all and only the fragments $t_{i,j}$ so that $t_{i,j} \in \mathcal{D}_L$. The algorithm exploits labels and production rules found in the fragments listed in the dictionary to generate only the fragments that *may be* in the dictionary. For example, if the dictionary does not contain a fragment whose root is labeled N, then if a node N is encountered during TFX neither its base fragment nor its expansions are generated. The process is applied to the whole training (*TFX-train*) and test (*TFX-test*) sets. The fragment space is now *explicit*, as there is a mapping between the input vectors and the fragments they encode.

3.4 Explicit Space Learning (*ESL*)

Linearized training data is used to learn a very fast model by using all the available data and a linear kernel.

3.5 Explicit Space Classification (*ESC*)

The linear model is used to classify linearized test data and evaluate the accuracy of the resulting classifier.

4 Previous work

A rather comprehensive overview of feature selection techniques is carried out in (Guyon and Elisseeff, 2003). Non-filter approaches for SVMs and kernel machines are often concerned with polynomial and Gaussian kernels, e.g. (Weston et al., 2001) and (Neumann et al., 2005). Weston et al. (2003) use the ℓ_0 norm in the SVM optimizer to stress the feature selection capabilities of the learning algorithm. In (Kudo and Matsumoto, 2003), an extension of the PrefixSpan algorithm (Pei et al., 2001) is used to efficiently mine the features in a low degree polynomial kernel space. The authors discuss an approximation of their method that allows them to handle high degree polynomial kernels.

Suzuki and Isozaki (2005) present an embedded approach to feature selection for convolution kernels based on χ^2-driven relevance assessment. To our knowledge, this is the only published work clearly focusing on feature selection for tree kernel functions, and indeed has been one of the major sources of inspiration for our methodology. With respect to their work, the difference

in our approach is that we want to exploit the SVM optimizer to select the most relevant features instead of a relevance assessment measure that moves from different statistical assumptions than the learning algorithm.

In (Graf et al., 2004), an approach to SVM parallelization is presented which is based on a divide-et-impera strategy to reduce optimization time. The idea of using a compact graph representation to represent the support vectors of a TK function is explored in (Aiolli et al., 2006), where a Direct Acyclic Graph (DAG) is employed. In (Moschitti, 2006; Bloehdorn and Moschitti, 2007a; Bloehdorn and Moschitti, 2007b; Moschitti et al., 2007), the SST kernel along with other tree and combined kernels are employed for question classification and semantic role labeling with interesting results.

5 Experiments

We evaluated the capability of our model to extract relevant features on two data sets: the CoNLL 2005 shared task on Semantic Role Labeling (SRL) (Carreras and Màrquez, 2005), and the Question Classification (QC) task based on data from the TREC 10 QA competition (Voorhees, 2001). The next sections will detail the setup and outcome of the two sets of experiments.

All the experiments were run on a machine equipped with 4 Intel® Xeon® CPUs clocked at 1.6 GHz and 4 GB of RAM. As a supervised learning framework we used SVM-Light-TK[1], which extends the SVM-Light optimizer (Joachims, 2000) with tree kernel support. For each classification task, we compare the accuracy of a vanilla SST classifier against the corresponding linearized SST classifier (SST$_\ell$). For KSL and SST training we used the default decay factor $\lambda = 0.4$. For ESL, we use a non-normalized, linear kernel. No further parametrization of the learning algorithms is carried out. Indeed, our focus is on showing that, under the same conditions, our linearized tree kernel can be as accurate as the original kernel, and choosing of parameters may just bias such test.

5.1 Semantic Role Labeling

For our experiments on semantic role labeling we used PropBank annotations (Palmer et al., 2005) and automatic Charniak parse trees (Charniak, 2000) as provided for the CoNLL 2005 evaluation campaign (Carreras and Màrquez, 2005). SRL can be decomposed into two tasks: *boundary detection*, where the word sequences that are arguments of a predicate word w are identified, and *role classification*, where each argument is assigned the proper role. The former task requires a binary *Boundary Classifier* (BC), whereas the second involves a *Role Multi-class Classifier* (RM).

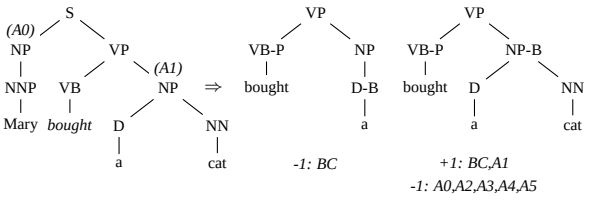


Figure 2: Examples of AST$_m$ structured features.

5.1.1 Setup

If the constituency parse tree t of a sentence s is available, we can look at all the pairs $\langle p, n_i \rangle$, where n_i is any node in the tree and p is the node dominating w, and decide whether n_i is an *argument node* or not, i.e. whether it exactly dominates all and only the words encoding any of w's arguments. The objects that we classify are subsets of the input parse tree that encompass both p and n_i. Namely, we use the AST$_m$ structure defined in (Moschitti et al., 2008), which is the minimal tree that covers all and only the words of p and n_i. In the AST$_m$, p and n_i are marked so that they can be distinguished from the other nodes. An AST$_m$ is regarded as a positive example for BC if n_i is an argument node, otherwise it is considered a negative example. Positive BC examples can be used to train an efficient RM: for each role r we can train a classifier whose positive examples are argument nodes whose label is exactly r, whereas negative examples are argument nodes labeled $r' \neq r$. Two AST$_m$s extracted from an example parse tree are shown in Figure 2: the first structure is a negative example for BC and is not part of the data set of RM, whereas the second is a positive instance for BC and A1.

To train BC we used PropBank sections 1 through 6, extracting AST$_m$ structures out of the first 1 million $\langle p, n_i \rangle$ pairs from the corresponding parse trees. As a test set we used the 149,140 instance collected from the annotations in Section 24. There are 61,062 positive examples in the training set (i.e. 6.1%) and 8,515 in the test set

[1] `http://disi.unitn.it/~moschitt/Tree-Kernel.htm`

(i.e. 5.7%).

For RM we considered all the argument nodes of any of the six PropBank core roles (i.e. A0, ..., A5) from all the available training sections, i.e. 2 through 21, for a total of 179,091 training instances. Similarly, we collected 5,928 test instances from the annotations of Section 24. Columns *Tr*$^+$ and *Te*$^+$ of Table 1 show the number of positive training and test examples, respectively, for BC and the role classifiers.

For all the linearized classifiers, we used 50 splits for the FMI stage and we set the threshold value $L = 50k$ and $maxexp = 1$ during FMI and TFX. We did not validate these parameters, which we know to be sub-optimal. These values were selected during the development of the software because, on a very small test bed, they resulted in a responsive and accurate system.

We should point out that other experiments have shown that linearization is very robust with respect to parametrization: due to the huge number and variety of fragments in the TK space, different choices of the parameters result in different explicit spaces and more or less efficient solutions, but in most cases the final accuracy of the linearized classifiers is affected only marginally. For example, it could be expected that reducing the number of splits during KSL would improve the final accuracy of a linearized classifier, as the weights used for FMI would then converge to the global optimum. Instead, we have observed that increasing the number of splits does not necessarily decrease the accuracy of the linearized classifier.

The evaluation on the whole SRL task using the official CoNLL'05 evaluator was not carried out because producing complete annotations requires several steps (e.g. overlap resolution, OvA or Pairwise combination of individual role classifiers) that would shade off the actual impact of the methodology on classification.

5.1.2 Results

The left side of Table 1 shows the distribution of positive data points in the training and test sets of each classifier. Columns *SST* and *SST*$_\ell$ compare side by side the F_1 measure of the non-linearized and linearized classifier for each class. The accuracy of the RM classifier is the percentage of correct class assignments.

We can see that the accuracy of linearized classifiers is always in line with vanilla SST, even if the selected linearization parameters generate a very rough approximation of the original fragment space, generally consisting of billions of fragments. BC$_\ell$ (i.e. the linearized BC) has an F_1 of 81.3, just 0.5% less than BC, i.e. 81.8. Concerning RM$_\ell$, its accuracy is the same as the non linearized classifier, i.e. 87.8.

	Data set		Accuracy	
Class	Tr$^+$	Te$^+$	SST	SST$_\ell$
BC	61,062	8,515	81.8	81.3
A0	60,900	2,014	91.6	91.1
A1	90,636	3,041	89.0	89.4
A2	21,291	697	73.1	73.0
A3	3,481	105	56.8	53.0
A4	2,713	69	69.1	67.9
A5	69	2	66.7	0.0
RM			87.8	87.8

Table 1: Number of positive training (Tr$^+$) and test (Te$^+$) examples in the SRL dataset. Accuracy of the non-linearized (SST) and linearized (SST$_\ell$) binary classifiers (i.e. BC, A0, ... A5) is F_1 measure. Accuracy of RM is the percentage of correct class assignments.

We should consider that the linearization framework can drastically improve the efficiency of learning and classification when dealing with large amounts of data. For a linearized classifier, we consider *training time* to be the overall time required to carry out the following activities: KSL, FMI, TFX on training data and ESL. Similarly, we consider test time the time necessary to perform TFX on test data and ESC. Training BC took more than two days of CPU time and testing about 4 hours, while training and testing the linearized boundary classifier required only 381 and 25 minutes, respectively. That is, on the same amount of data we can train a linearized classifier about 8 times as fast, and test it in about 1 tenth of the time. Concerning RM, sequential training of the 6 models took 2,596 minutes, while testing took 27 minutes. The linearized role multi classifier required 448 and 24 minutes for training and testing, respectively, i.e. training is about 5 times as fast while testing time is about the same. If compared with the boundary classifier, the improvement in efficiency is less evident: indeed, the relatively small size of the role classifiers data sets limits the positive effect of splitting training data into smaller chunks.

SRL fragment space. Table 3 lists the best fragments identified for the Boundary Classifier. We should remember that we are using AST$_m$ struc-

tures as input to our classifiers: nodes whose label end with "-P" are predicate nodes, while nodes whose label ends with "-B" are candidate argument nodes.

All the most relevant fragments encode the minimum sub-tree encompassing the predicate and the argument node. This kind of structured feature subsumes several features traditionally employed for explicit SRL models: the Path (i.e. the sequence of nodes connecting the predicate and the candidate argument node), Phrase Type (i.e. the label of the candidate argument node), Predicate POS (i.e. the POS of the predicate word), Position (i.e. whether the argument is to the left or to the right of the predicate) and Governing Category (i.e. the label of the common ancestor) defined in (Gildea and Jurafsky, 2002).

The linearized model for BC contains about 160 thousand fragments. Of these, about 70 and 33 thousand encompass the candidate argument or the predicate node, respectively. About 16 thousand fragments contain both.

5.2 Question Classification

For question classification we used the data set from the TREC 10 QA evaluation campaign[2], consisting of 5,500 training and 500 test questions.

5.2.1 Setup

Given a question, the QC task consists in selecting the most appropriate expected answer type from a given set of possibilities. We adopted the question taxonomy known as *coarse grained*, which has been described in (Zhang and Lee, 2003) and (Li and Roth, 2006), consisting of six non overlapping classes: Abbreviations (ABBR), Descriptions (DESC, e.g. definitions or explanations), Entity (ENTY, e.g. animal, body or color), Human (HUM, e.g. group or individual), Location (LOC, e.g. cities or countries) and Numeric (NUM, e.g. amounts or dates).

For each question, we generate the full parse of the sentence and use it to train SST and (linearized) SST_ℓ models. The automatic parses are obtained with the Stanford parser[3] (Klein and Manning, 2003). We actually have only 5,483 sentences in our training set, due to parsing issues with a few of them.

[2] `http://l2r.cs.uiuc.edu/cogcomp/Data/QA/QC/`

[3] `http://nlp.stanford.edu/software/lex-parser.shtml`

Class	Data set Tr^+	Te^+	Accuracy SST	SST_ℓ
ABBR	89	9	80.0	87.5
DESC	1,164	138	96.0	94.5
ENTY	1,269	94	63.9	63.5
HUM	1,231	65	88.1	87.2
LOC	834	81	77.6	77.9
NUM	896	113	80.4	80.8
Overall			86.2	86.6

Table 2: Number of positive training (Tr^+) and test (Te^+) examples in the QA dataset. Accuracy of the non-linearized (SST) and linearized (SST_ℓ) binary classifiers is F_1 measure. Overall accuracy is the percentage of correct class assignments.

The classifiers are arranged in a one-vs.-all (OvA) configuration, where each sentence is a positive example for one of the six classes, and negative for the other five. Given the very small size of the data set, we used $S = 1$ during KSL for the linearized classifier (i.e. we didn't partition training data). We carried out no validation of the parameters, and we used $maxexp = 4$ and $L = 50k$ in order to generate a rich fragment space.

5.2.2 Results

Table 2 shows the number of positive examples in the training and test set of each individual binary classifiers. Columns SST and SST_ℓ compare the F_1 measure of the vanilla and linearized classifiers on the individual classes, and the accuracy of the complete QC task (Row *Overall*) in terms of percentage of correct class assignments. Also in this case, we can notice that the accuracy of the linearized classifiers is always in line with non-linearized ones, e.g. 86.6 vs. 86.2 for the multi-classifiers. These results are lower than those derived in (Moschitti, 2006; Moschitti et al., 2007), i.e. 88.2 and 90.4, respectively, where the parameters for each classifier were carefully optimized.

QC Fragment space. Tables from 4 to 9 list the top fragments identified for each class [4].

As expected, for all the categories the domain lexical information is very relevant. For example, *film*, *color*, *book*, *novel* and *sport* for ENTY or *city*, *country*, *state* and *capital* for LOC. Of the six classes, ENTY (Table 6) is mostly characterized by lexical features. Interestingly, function words, which would have been eliminated by a pure Information Retrieval approach (i.e. by means of

[4] Some categories show meaningful syntactic fragments after the first 10, so for them we report more subtrees.

standard stop-list), are in the top positions, e.g.: *why* and *how* for DESC, *what* for ENTY, *who* for HUM, *where* for LOC and *when* for NUM. For the latter, also *how* seems to be important suggesting that features may strongly characterize more than one given class.

Characteristic syntactic features appear in the top positions for each class, for example: *(VP (VB (stand)) (PP))*, which suggests that *stand* should be followed by a prepositional phrase to characterize ABBR; or *(NP (NP (DT) (NN (abbreviation))) (PP))*, which suggests that, to be in a relevant pattern, *abbreviation* should be preceded by an article and followed by a PP. Also, the syntactic structure is useful to differentiate the use of the same important words, e.g. *(SBARQ (WHADVP (WRB (How))) (SQ) (.))* for DESC better characterizes the use of *how* with respect to NUM, in which a relevant use is *(WHADJP (WRB (How)) (JJ))*.

In (Moschitti et al., 2007) it was shown that the use of TK improves QC of 1.2 percent points, i.e. from 90.6 to 91.8: further analysis of these fragments may help us to device compact, less sparse syntactic features and design more accurate models for the task.

6 Discussion

The fact that our model doesn't always improve the accuracy of a standard SST model might be related to the process of splitting training data and employing locally estimated weights during FMI.

Concerning the experiments presented in this paper, this objection might apply to the results on SRL, where we used 50 splits to identify the most relevant fragments, but not to those on QC, where given the limited size of the data set we decided not to split training data at all as explained in Section 5.2. Furthermore, as we already discussed, we have evidence that there is no direct correlation between the number of splits used for KSL and the accuracy of the resulting classifier. After all, the optimization carried out during ESL is global, and we can assume that, if we mined enough fragments during FMI, than those actually retained by the global linear model would be by and large the same, regardless of the split configuration.

More in general, feature selection may give an improvement to some learning algorithm but if it can help SVMs is debatable, since its related theory show that they are robust to irrelevant features. In our specific case, we remove features

(ADJP(RB-B)(VBN-P))
(NP(VBN-P)(NNS-B))
(S(NP-B)(VP))
(VP(VBD-P(said))(SBAR))
(VP(VB-P)(NP-B))
(NP(VBG-P)(NNS-B))
(VP(VBD-P)(NP-B))
(VP(VBG-P)(NP-B))
(VP(VBZ-P)(NP-B))
(VP(VBN-P)(NP-B))
(VP(VBP-P)(NP-B))
(NP(NP-B)(VP))
(NP(VBG-P)(NN-B))
(S(S(VP(VBG-P)))(NP-B))

Table 3: Best fragments for SRL BC.

(NN(abbreviation))
(NP(DT)(NN(abbreviation)))
(NP(DT(the))(NN(abbreviation)))
(IN(for))
(VB(stand))
(VBZ(does))
(PP(IN))
(VP(VB(stand))(PP))
(NP(NP(DT)(NN(abbreviation)))(PP))
(SQ(VBZ)(NP)(VP(VB(stand))(PP)))
(SBARQ(WHNP)(SQ(VBZ)(NP)(VP(VB(stand))(PP)))(.))
(SQ(VBZ(does))(NP)(VP(VB(stand))(PP)))
(VP(VBZ)(NP(NP(DT)(NN(abbreviation)))(PP)))

Table 4: Best fragments for the ABBR class.

(WRB(Why))
(WHADVP(WRB(Why)))
(WHADVP(WRB(How)))
(WHADVP(WRB))
(VB(mean))
(VBZ(causes))
(VB(do))
(ROOT(SBARQ(WHADVP(WRB(How)))(SQ)(.)))
(ROOT(SBARQ(WHADVP(WRB(How)))(SQ)(.(?))))
(SBARQ(WHADVP(WRB(How)))(SQ))
(WRB(How))
(SBARQ(WHADVP(WRB(How)))(SQ)(.))
(SBARQ(WHADVP(WRB(How)))(SQ)(.(?)))
(SBARQ(WHADVP(WRB(Why)))(SQ))
(ROOT(SBARQ(WHADVP(WRB(Why)))(SQ)))
(SBARQ(WHADVP(WRB))(SQ))

Table 5: Best fragments for the DESC class.

(NN(film))
(NN(color))
(NN(book))
(NN(novel))
(NN(sport))
(WP(What))
(NN(fear))
(NN(movie))
(NN(word))
(VP(VBN(called)))
(NN(game))
(NP(DT)(NN(fear)))
(NP(NP(DT)(NN(fear)))(PP))

Table 6: Best fragments for the ENTY class.

(NN(company))
(WP(Who))
(WHNP(WP(Who)))
(NN(name))
(NN(team))
(NN(baseball))
(WHNP(WP))
(NN(character))
(NNP(President))
(NN(leader))
(NN(actor))
(NN(president))
(JJ(Whose))
(VP(VBD)(NP))
(NP(NP)(JJ)(NN(name)))
(VP(VBD)(VP))
(NN(organization))
(VP(VBD)(NP)(PP(IN)(NP)))
(SBARQ(WHNP(WP(Who)))(SQ)(.))
(ROOT(SBARQ(WHNP(WP(Who)))(SQ)(.)))
(ROOT(SBARQ(WHNP(WP(Who)))(SQ)(.(?))))
(SBARQ(WHNP(WP(Who)))(SQ)(.(?)))

Table 7: Best fragments for the HUM class.

(NN(city))
(NN(country))
(WRB(Where))
(NN(state))
(WHADVP(WRB(Where)))
(NN(capital))
(NP(NN(city)))
(NNS(countries))
(NP(NN(state)))
(PP(IN(in)))
(SBARQ(WHADVP(WRB(Where)))(SQ)(.(?)))
(SBARQ(WHADVP(WRB(Where)))(SQ)(.))
(ROOT(SBARQ(WHADVP(WRB(Where)))(SQ)(.)))
(ROOT(SBARQ(WHADVP(WRB(Where)))(SQ)(.(?))))
(NN(island))
(NN(address))
(NN(river))
(NN(mountain))
(ROOT(SBARQ(WHADVP(WRB(Where)))(SQ)))
(SBARQ(WHADVP(WRB(Where)))(SQ))

Table 8: Best fragments for the LOC class.

(WRB(How))
(WHADVP(WRB(When)))
(WRB(When))
(JJ(many))
(NN(year))
(WHADJP(WRB)(JJ))
(NP(NN(year)))
(WHADJP(WRB(How))(JJ))
(NN(date))
(SBARQ(WHADVP(WRB(When)))(SQ)(.(?)))
(SBARQ(WHADVP(WRB(When)))(SQ)(.))
(NN(day))
(NN(population))
(ROOT(SBARQ(WHADVP(WRB(When)))(SQ)(.)))
(ROOT(SBARQ(WHADVP(WRB(When)))(SQ)(.(?))))
(JJ(average))
(NN(number))

Table 9: Best fragments for the NUM class.

whose SVM weights are the lowest, i.e. those that are (almost) irrelevant for the SVM. Therefore, the chance of this resulting in an improvement is rather low.

With respect to cases where our model is less accurate than a standard SST, we should consider that our choice of parameters is sub-optimal and we adopt a *very* aggressive feature selection strategy, that only retains a few thousand features from a space where there are hundreds of millions of different features.

7 Conclusions

We introduced a novel framework for support vector classification that combines advantages of convolution kernels, i.e. the generation of a very high dimensional structure space, with the efficiency and clarity of explicit representations in a linear space.

For this paper, we focused on the SubSet Tree kernel and verified the potential of the proposed solution on two NLP tasks, i.e. semantic role labeling and question classification. The experiments show that our framework drastically reduces processing time, e.g. boundary classification for SRL, while preserving the accuracy.

We presented a selection of the most relevant fragments identified for the SRL boundary classifier as well as for each class of the coarse grained QC task. Our analysis shows that our framework can discover state-of-the-art features, e.g. the Path feature for SRL. We believe that sharing these fragments with the NLP community and studying them in more depth will be useful to identify new, relevant features for the characterization of several learning problems. For this purpose, we made available the fragment spaces at `http://danielepighin.net` and we will keep them updated with new set of experiments on new tasks, e.g. SRL based on FrameNet and VerbNet, e.g. (Giuglea and Moschitti, 2004).

In our future work, we plan to widen the list of covered tasks and to extend our algorithm to cope with different kernel families, such as the partial tree kernel and kernels defined over pairs of trees, e.g. the ones used for textual entailment in (Moschitti and Zanzotto, 2007). We also plan to move from mining fragments to mining classes of fragments, i.e. to identify prototypical fragments in the fragment space that generalize topological sub-classes of the most relevant fragments.

References

Fabio Aiolli, Giovanni Da San Martino, Alessandro Sperduti, and Alessandro Moschitti. 2006. Fast on-line kernel learning for trees. In *Proceedings of ICDM'06*.

Stephan Bloehdorn and Alessandro Moschitti. 2007a. Combined syntactic and semantic kernels for text classification. In *Proceedings of ECIR 2007, Rome, Italy*.

Stephan Bloehdorn and Alessandro Moschitti. 2007b. Structure and semantics for expressive text kernels. In *In Proceedings of CIKM '07*.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. 2003. Word sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL'05*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL'00*.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of ACL'02*.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of ACL'04*.

Chad Cumby and Dan Roth. 2003. Kernel Methods for Relational Learning. In *Proceedings of ICML 2003*.

Mona Diab, Alessandro Moschitti, and Daniele Pighin. 2008. Semantic role labeling systems for Arabic using kernel methods. In *Proceedings of ACL-08: HLT*, pages 798–806.

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

Ana-Maria Giuglea and Alessandro Moschitti. 2004. Knowledge discovery using framenet, verbnet and propbank. In A. Meyers, editor, *Workshop on Ontology and Knowledge Discovering at ECML 2004*, Pisa, Italy.

Alfio Gliozzo, Claudio Giuliano, and Carlo Strapparava. 2005. Domain kernels for word sense disambiguation. In *Proceedings of ACL'05*, pages 403–410.

Hans P. Graf, Eric Cosatto, Leon Bottou, Igor Durdanovic, and Vladimir Vapnik. 2004. Parallel support vector machines: The cascade svm. In *Neural Information Processing Systems*.

Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

David Haussler. 1999. Convolution kernels on discrete structures. Technical report, Dept. of Computer Science, University of California at Santa Cruz.

T. Joachims. 2000. Estimating the generalization performance of a SVM efficiently. In *Proceedings of ICML'00*.

Hisashi Kashima and Teruo Koyanagi. 2002. Kernels for semi-structured data. In *Proceedings of ICML'02*.

Jun'ichi Kazama and Kentaro Torisawa. 2005. Speeding up training with tree kernels for node relation labeling. In *Proceedings of HLT-EMNLP'05*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL'03*, pages 423–430.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of ACL'03*.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proceedings of ACL'05*.

Xin Li and Dan Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249.

Alessandro Moschitti and Fabio Massimo Zanzotto. 2007. Fast and effective kernels for relational learning from texts. In *ICML'07*.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proceedings of ACL'07*.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of ECML'06*, pages 318–329.

Julia Neumann, Christoph Schnorr, and Gabriele Steidl. 2005. Combined SVM-Based Feature Selection and Classification. *Machine Learning*, 61(1-3):129–150.

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

J. Pei, J. Han, Mortazavi B. Asl, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu. 2001. PrefixSpan Mining Sequential Patterns Efficiently by Prefix Projected Pattern Growth. In *Proceedings of ICDE'01*.

Daniele Pighin and Alessandro Moschitti. 2009. Efficient linearization of tree kernel functions. In *Proceedings of CoNLL'09*.

Alain Rakotomamonjy. 2003. Variable selection using SVM based criteria. *Journal of Machine Learning Research*, 3:1357–1370.

Libin Shen, Anoop Sarkar, and Aravind k. Joshi. 2003. Using LTAG Based Features in Parse Reranking. In *Proceedings of EMNLP'06*.

Jun Suzuki and Hideki Isozaki. 2005. Sequence and Tree Kernels with Statistical Feature Mining. In *Proceedings of NIPS'05*.

Kristina Toutanova, Penka Markova, and Christopher Manning. 2004. The Leaf Path Projection View of Parse Trees: Exploring String Kernels for HPSG Parse Selection. In *Proceedings of EMNLP 2004*.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

Ellen M. Voorhees. 2001. Overview of the trec 2001 question answering track. In *In Proceedings of the Tenth Text REtrieval Conference (TREC*, pages 42–51.

Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. 2001. Feature Selection for SVMs. In *Proceedings of NIPS'01*.

Jason Weston, André Elisseeff, Bernhard Schölkopf, and Mike Tipping. 2003. Use of the zero norm with linear models and kernel methods. *J. Mach. Learn. Res.*, 3:1439–1461.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *Proceedings of SIGIR'03*, pages 26–32.

A Rich Feature Vector for Protein-Protein Interaction Extraction from Multiple Corpora

Makoto Miwa[1] **Rune Sætre**[1] **Yusuke Miyao**[1] **Jun'ichi Tsujii**[1,2,3]
[1]Department of Computer Science, the University of Tokyo, Japan
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan.
[2]School of Computer Science, University of Manchester, UK
[3]National Center for Text Mining, UK
{mmiwa,rune.saetre,yusuke,tsujii}@is.s.u-tokyo.ac.jp

Abstract

Because of the importance of protein-protein interaction (PPI) extraction from text, many corpora have been proposed with slightly differing definitions of proteins and PPI. Since no single corpus is large enough to saturate a machine learning system, it is necessary to learn from multiple different corpora. In this paper, we propose a solution to this challenge. We designed a rich feature vector, and we applied a support vector machine modified for corpus weighting (SVM-CW) to complete the task of multiple corpora PPI extraction. The rich feature vector, made from multiple useful kernels, is used to express the important information for PPI extraction, and the system with our feature vector was shown to be both faster and more accurate than the original kernel-based system, even when using just a single corpus. SVM-CW learns from one corpus, while using other corpora for support. SVM-CW is simple, but it is more effective than other methods that have been successfully applied to other NLP tasks earlier. With the feature vector and SVM-CW, our system achieved the best performance among all state-of-the-art PPI extraction systems reported so far.

1 Introduction

The performance of an information extraction program is highly dependent on various factors, including text types (abstracts, complete articles, reports, etc.), exact definitions of the information to be extracted, shared sub-topics of the text collections from which information is to be extracted. Even if two corpora are annotated in terms of the same type of information by two groups, the performance of a program trained by one corpus is unlikely to be reproduced in the other corpus. On the other hand, from a practical point of view, it is worth while to effectively use multiple existing annotated corpora together, because it is very costly to make new annotations.

One problem with several different corpora is protein-protein interaction (PPI) extraction from text. While PPIs play a critical role in understanding the working of cells in diverse biological contexts, the manual construction of PPI databases such as BIND, DIP, HPRD, IntAct, and MINT (Mathivanan et al., 2006) is known to be very time-consuming and labor-intensive. The automatic extraction of PPI from published papers has therefore been a major research topic in Natural Language Processing for Biology (BioNLP).

Among several PPI extraction task settings, the most common is sentence-based, pair-wise PPI extraction. At least four annotated corpora have been provided for this setting: AIMed (Bunescu et al., 2005), HPRD50 (Fundel et al., 2006), IEPA (Ding et al., 2002), and LLL (Nédellec, 2005). Each of these corpora have been used as the standard corpus for training and testing PPI programs. Moreover, several corpora are annotated for more types of events than just for PPI. Such examples include BioInfer (Pyysalo et al., 2007), and GENIA (Kim et al., 2008a), and they can be reorganized into PPI corpora. Even though all of these corpora were made for PPI extraction, they were constructed based on different definitions of proteins and PPI, which reflect different biological research interests (Pyysalo et al., 2008).

Research on PPI extraction so far has revealed that the performance on each of the corpora could

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 121–130,
Singapore, 6-7 August 2009.

benefit from additional examples (Airola et al., 2008). Learning from multiple annotated corpora could lead to better PPI extraction performance. Various research paradigms such as inductive transfer learning (ITL) and domain adaptation (DA) have mainly focused on how to effectively use corpora annotated by other groups, by reducing the incompatibilities (Pan and Yang, 2008).

In this paper, we propose the extraction of PPIs from multiple different corpora. We design a rich feature vector, and as an ITL method, we apply a support vector machine (SVM) modified for corpus weighting (SVM-CW) (Schweikert et al., 2008), in order to evaluate the use of multiple corpora for the PPI extraction task. Our rich feature vector is made from multiple useful kernels, each of which is based on multiple parser inputs, proposed by Miwa et al. (2008). The system with our feature vector was better than or at least comparable to the state-of-the-art PPI extraction systems on every corpus. The system is a good starting point to use the multiple corpora. Using one of the corpora as the target corpus, SVM-CW weights the remaining corpora (we call them the source corpora) with "goodness" for training on the target corpus. While SVM-CW is simple, we show that SVM-CW can improve the performance of the system more effectively and more efficiently than other methods proven to be successful in other NLP tasks earlier. As a result, SVM-CW with our feature vector is comprised of a PPI system with five different models, of which each model is superior to the best model in the original PPI extraction task, which used only the single corpus.

2 Related Works

While sentence-based, pair-wise PPI extraction was initially tackled by using simple methods based on co-occurrences, lately, more sophisticated machine learning systems augmented by NLP techniques have been applied (Bunescu et al., 2005). The task has been tackled as a classification problem. To pull out useful information from NLP tools including taggers and parsers, several kernels have been applied to calculate the similarity between PPI pairs. Miwa et al. (2008) recently proposed the use of multiple kernels using multiple parsers. This outperformed other systems on the AIMed, which is the most frequently used corpus for the PPI extraction task, by a wide margin.

To improve the performance using external

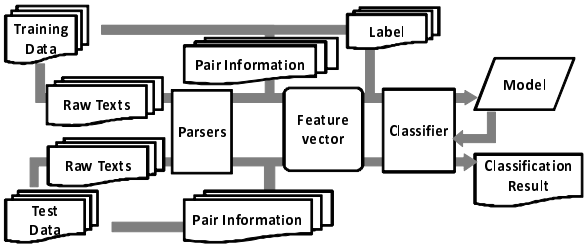


Figure 1: Overview of our PPI extraction system

training data, many ITL and DA methods have been proposed. Most of ITL methods assume that the feature space is same, and that the labels may be different in only some examples, while most of DA methods assume that the labels are the same, and that the feature space is different. Among the methods, we use adaptive SVM (aSVM) (Yang et al., 2007), singular value decomposition (SVD) based alternating structure optimization (SVD-ASO) (Ando et al., 2005), and transfer AdaBoost (TrAdaBoost) (Dai et al., 2007) to compare with SVM-CW. We do not use semi-supervised learning (SSL) methods, because it would be considerably costly to generate enough clean unlabeled data needed for SSL (Erkan et al., 2007). aSVM is seen as a promising DA method among several modifications of SVM including SVM-CW. aSVM tries to find a model that is close to the one made from other classification problems. SVD-ASO is one of the most successful SSL, DA, or multi-task learning methods in NLP. The method tries to find an additional useful feature space by solving auxiliary problems that are close to the target problem. With well-designed auxiliary problems, the method has been applied to text classification, text chunking, and word sense disambiguation (Ando, 2006). The method was reported to perform better than or comparable to the best state-of-the-art systems in all of these tasks. TrAdaBoost was proposed as an ITL method. In training, the method reduces the effect of incompatible examples by decreasing their weights, and thereby tries to use useful examples from source corpora. The method has been applied to text classification, and the reported performance was better than SVM and transductive SVM (Dai et al., 2007).

3 PPI Extraction System

The target task of our system is a sentence-based, pair-wise PPI extraction. It is formulated as a classification problem that judges whether a given pair

XPG$_{p1}$ protein interacts with multiple subunits of **TFIIH**$_{prot}$ and with **CSB**$_{p2}$ protein.

Figure 2: A sentence including an interacting protein pair (p1, p2). (AIMed PMID 8652557, 9th sentence, 3rd pair)

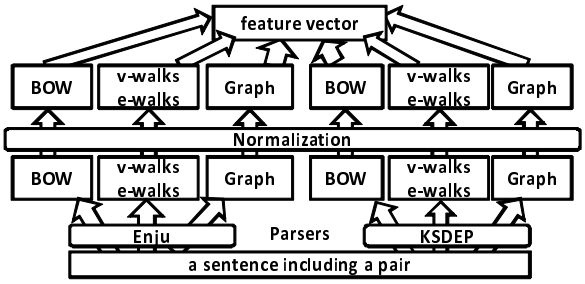


Figure 3: Extraction of a feature vector from the target sentence

of proteins in a sentence is interacting or not. Figure 2 shows an example of a sentence in which the given pair (p1 and p2) actually interacts.

Figure 1 shows the overview of the proposed PPI extraction system. As a classifier using a single corpus, we use the 2-norm soft-margin linear SVM (L2-SVM) classifier, with the dual coordinate decent (DCD) method, by Hsieh et al. (2008). In this section, we explain the two main features: the feature vector, and the corpus weighting method for multiple corpora.

3.1 Feature Vector

We propose a feature vector with three types of features, corresponding to the three different kernels, which were each combined with the two parsers: the Enju 2.3.0, and KSDEP beta 1 (Miyao et al., 2008); this feature vector is used because the kernels with these parsers were shown to be effective for PPI extraction by Miwa et al. (2008), and because it is important to start from a good performance single corpus system. Both parsers were retrained using the GENIA Treebank corpus provided by Kim et al. (2003). By using our linear feature vector, we can perform calculations faster by using fast linear classifiers like L2-SVM, and we also obtain a more accurate extraction, than by using the original kernel method.

Figure 3 summarizes the way in which the feature vector is constructed. The system extracts Bag-of-Words (BOW), shortest path (SP), and graph features from the output of two parsers. The

PROT_M:1, and_M:1, interact_M:1, multiple_M:1, of_M:1, protein_M:1, subunit_M:1, with_M:2, protein_A:1

Figure 4: Bag-of-Words features of the pair in Figure 2 with their positions (B:Before, M:in the Middle of, A:After) and frequencies.

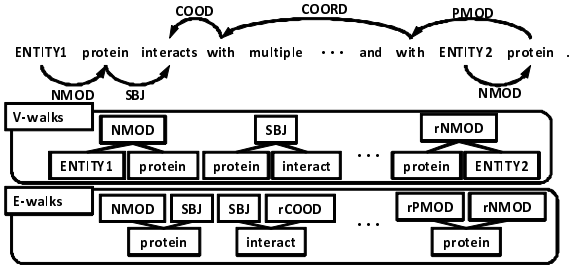


Figure 5: Vertex walks, edge walks in the upper shortest path between the proteins in the parse tree by KSDEP. The walks and their subsets are used as the shortest path features of the pair in Figure 2.

output is grouped according to the feature-type and parser, and each group of features is separately normalized by the L2-norm[1]. Finally, all values are put into a single feature vector, and the whole feature vector is then also normalized by the L2-norm. The features are constructed by using predicate argument structures (PAS) from Enju, and by using the dependency trees from KSDEP.

3.1.1 Bag-of-Words (BOW) Features

The BOW feature includes the lemma form of a word, its relative position to the target pair of proteins (Before, Middle, After), and its frequency in the target sentence. BOW features form the BOW kernel in the original kernel method. BOW features for the pair in Figure 2 are shown in Figure 4.

3.1.2 Shortest Path (SP) Features

SP features include vertex walks (v-walks), edge walks (e-walks), and their subsets (Kim et al., 2008b) on the target pair in a parse structure, and represent the connection between the pair. The features are the subsets of the tree kernels on the shortest path (Sætre et al., 2007). Figure 5 illustrates the shortest path between the pair in Figure 2, and its v-walks and e-walks extracted from the shortest path in the parse tree by KSDEP. A v-walk includes two lemmas and their link, while

[1]The vector normalized by the L2-norm is also called a unit vector.

an e-walk includes a lemma and its two links. The links indicates the predicate argument relations for PAS, and the dependencies for dependency trees.

3.1.3 Graph Features

Graph features are made from the all-paths graph kernel proposed by Airola et al. (2008). The kernel represents the target pair using graph matrices based on two subgraphs, and the graph features are all the non-zero elements in the graph matrices.

The two subgraphs are a parse structure subgraph (PSS) and a linear order subgraph (LOS). Figure 6 describes the subgraphs of the sentence parsed by KSDEP in Figure 2. PSS represents the parse structure of a sentence. PSS has word vertices or link vertices. A word vertex contains its lemma and its part-of-speech (POS), while a link vertex contains its link. Additionally, both types of vertices contain their positions relative to the shortest path. The "IP"s in the vertices on the shortest path represent the positions, and the vertices are differentiated from the other vertices like "P", "CC", and "and:CC" in Figure 6. LOS represents the word sequence in the sentence. LOS has word vertices, each of which contains its lemma, its relative position to the target pair, and its POS.

Each subgraph is represented by a graph matrix $\boldsymbol{G}$ as follows:

$$\boldsymbol{G} = \boldsymbol{L}^T \sum_{n=1}^{\infty} \boldsymbol{A}^n \boldsymbol{L}, \qquad (1)$$

where $\boldsymbol{L}$ is a $N \times L$ label matrix, $\boldsymbol{A}$ is an $N \times N$ edge matrix, N represents the number of vertices, and L represents the number of labels. The label of a vertex includes all information described above (e.g. "ENTITY1:NN:IP" in Figure 6). If two vertices have exactly same information, the labels will be same. $\boldsymbol{G}$ can be calculated efficiently by using the Neumann Series (Airola et al., 2008). The label matrix represents the correspondence between labels and vertices. $\boldsymbol{L}_{ij}$ is 1 if the i-th vertex corresponds to the j-th label, and 0 otherwise. The edge matrix represents the connection between the pairs of vertices. $\boldsymbol{A}_{ij}$ is a weight w_{ij} (0.9 or 0.3 in Figure 6 (Airola et al., 2008)) if the i-th vertex is connected to the j-th vertex, and 0 otherwise. By this calculation, $\boldsymbol{G}_{ij}$ represent the sum of the weights of all paths between the i-th label and the j-th label.

	A	B	H	I	L
positive	1,000	2,534	163	335	164
all	5,834	9,653	433	817	330

Table 1: The sizes of used PPI corpora. A:AIMed, B:BioInfer, H:HPRD50, I:IEPA, and L:LLL.

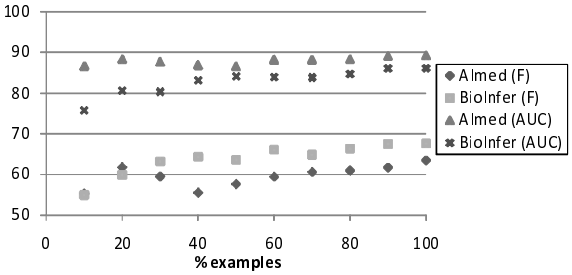


Figure 7: Learning curves on two large corpora. The x-axis is related to the percentage of the examples in a corpus. The curves are obtained by a 10-fold CV with a random split.

3.2 Corpus Weighting for Mixing Corpora

Table 1 shows the sizes of the PPI corpora that we used. Their widely-ranged differences including the sizes were manually analyzed by Pyysalo et al. (2008). While AIMed, HPRD50, IEPA, and LLL were all annotated as PPI corpora, BioInfer in its original form contains much more fine-grained information than does just the PPI. BioInfer was transformed into a PPI corpus by a program, so making it the largest of the five. Among them, AIMed alone was created by annotating whole abstracts, while the other corpora were made by annotating single sentences selected from abstracts.

Figure 7 shows the learning curves on two large corpora: AIMed and BioInfer. The curves are obtained by performing a 10-fold cross validation (CV) on each corpus, with random splits, using our system. The curves show that the performances can benefit from the additional examples. To get a better PPI extraction system for a chosen target, we need to draw useful shared information from external source corpora. We refer to examples in the source corpora as "source examples", and examples in a target corpus as "target examples". Among the corpora, we assume that the labels in some examples are incompatible, and that their distributions are also different, but that the feature space is shared.

In order to draw useful information from the source corpora to get a better model for the target

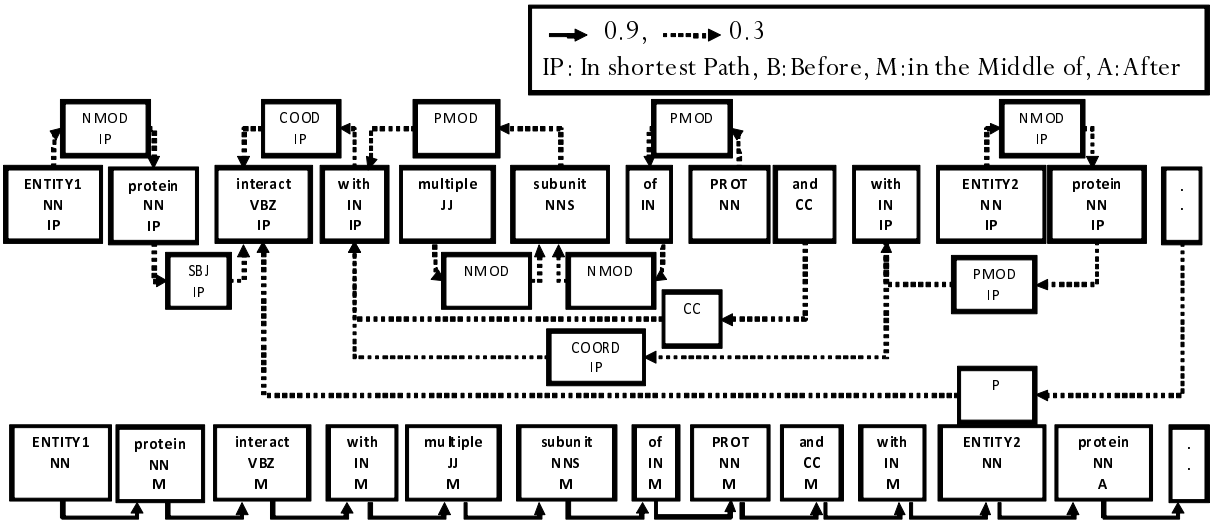


Figure 6: Parse structure subgraph and linear order subgraph to extract graph features of the pair in Figure 2. The parse structure subgraph is from the parse tree by KSDEP.

corpus, we use SVM-CW, which has been used as a DA method. Given a set of instance-label pairs ($\boldsymbol{x_i}$, y_i), $i = 1, \ldots, ls + lt$, $\boldsymbol{x_i} \in R^n$, and $y_i \in \{-1, +1\}$, we solve the following problem:

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C_s \sum_{i=1}^{ls} \ell_i + C_t \sum_{i=ls+1}^{ls+lt} \ell_i, \quad (2)$$

where $\boldsymbol{w}$ is a weight vector, ℓ is a loss function, and ls and lt are the numbers of source and target examples respectively. $C_s \geq 0$ and $C_t \geq 0$ are penalty parameters. We use a squared hinge loss $\ell_i = max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x_i})^2$. Here, the source corpora are treated as one corpus. The problem, excluding the second term, is equal to L2-SVM. The problem can be solved using the DCD method.

As an ITL method, SVM-CW weights each corpus, and tries to benefit from the source corpora, by adjusting the effect of their compatibility and incompatibility. For the adjustment, these penalty parameters should be set properly. Since we are unaware of the widely ranged differences among the corpora, we empirically estimated them by performing 10-fold CV on the training data.

4 Evaluation

4.1 Evaluation Settings

We used five corpora for evaluation: AIMed, BioInfer, HPRD50, IEPA, and LLL. For the comparison with other methods, we report the F-score (%), and the area under the receiver operating characteristic (ROC) curve (AUC) (%) using (abstract-wise) a 10-fold CV and a one-answer-per-occurrence criterion. These measures are commonly used for the PPI extraction tasks. The F-score is a harmonic mean of Precision and Recall. The ROC curve is a plot of a true positive rate (TPR) vs a false positive rate (FPR) for different thresholds. We tuned the regularization parameters of all classifiers by performing a 10-fold CV on the training data using a random split. The other parameters were fixed, and we report the highest of the macro-averaged F-scores as our final F-score. For 10-fold CV, we split the corpora as recommended by Airola et al. (2008).

4.2 PPI Extraction on a Single Corpus

In this section, we evaluate our system on a single corpus, in order to evaluate our feature vector and to justify the use of the following modules: normalization methods and classification methods.

First, we compare our preprocessing method with other preprocessing methods to confirm how our preprocessing method improves the performance. Our method produced 64.2% in F-score using L2-SVM on AIMed. Scaling all features individually to have a maximal absolute value of 1, produced only 44.2% in the F-score, while normalizing the feature vector by L2-norm produced 61.5% in the F-score. Both methods were inferior to our method, because the values of features in the same group should be treated together, and because the values of features in the different groups should not have a big discrepancy. Weighting each

	L2	L1	LR	AP	CW
F	**64.2**	64.0	**64.2**	62.7	63.0
AUC	**89.1**	88.8	89.0	88.5	87.8

Table 2: Classification performance on AIMed using five different linear classifiers. The F-score (F) and Area Under the ROC curve (AUC) are shown. L2 is L2-SVM, L1 is L1-SVM, LR is logistic regression, AP is averaged perceptron, and CW is confidence weighted linear classification.

group with different values can produce better results, as will be explored in our future work.

Next, using our feature vector, we applied five different linear classifiers to extract PPI from AIMed: L2-SVM, 1-norm soft-margin SVM (L1-SVM), logistic regression (LR) (Fan et al., 2008), averaged perceptron (AP) (Collins, 2002), and confidence weighted linear classification (CW) (Dredze et al., 2008). Table 2 indicates the performance of these classifiers on AIMed. We employed better settings for the task than did the original methods for AP and CW. We used a Widrow-Hoff learning rule (Bishop, 1995) for AP, and we performed one iteration for CW. L2-SVM is as good as, if not better, than other classifiers (F-score and AUC). In the least, L2-SVM is as fast as these classifiers. AP and CW are worse than the other three methods, because they require a large number of examples, and are un-suitable for the current task. This result indicates that all linear classifiers, with the exception of AP and CW, perform almost equally, when using our feature vector.

Finally, we implemented the kernel method by Miwa et al. (2008). For a 10-fold CV on AIMed, the running time was 9,507 seconds, and the performance was 61.5% F-score and 87.1% AUC. Our system used 4,702 seconds, and the performance was 64.2% F-score and 89.1% AUC. This result displayed that our system, with L2-SVM, and our new feature vector, is better, and faster, than the kernel-based system.

4.3 Evaluation of Corpus Weighting

In this section, we first apply each model from a source corpus to a target corpus, to show how different the corpora are. We then evaluate SVM-CW by comparing it with three other methods (see Section 2) with limited features, and apply it to every corpus.

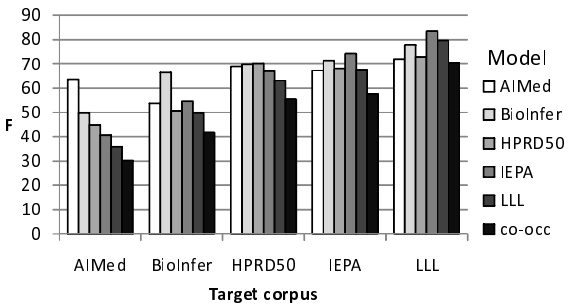


Figure 8: F-score on a target corpus using a model on a source corpus. For the comparison, we show the 10-fold CV result on each target corpus and co-occurrences. The regularization parameter was fixed to 1.

First, we apply the model from a source corpus to a target corpus. Figure 8 shows how the model from a source corpus performs on the target corpus. Interestingly, the model from IEPA performs better on LLL than the model from LLL itself. All the results showed that using different corpora (except IEPA) is worse than just using the same corpora. However, the cross-corpora scores are still better than the co-occurrences base-line, which indicates that the corpora share some information, even though they are not fully compatible.

Next, we compare SVM-CW with three other methods: aSVM, SVD-ASO, and TrAdaBoost. For this comparison, we used our feature vector without including the graph features, because SVD-ASO and TrAdaBoost require large computational resources. We applied SVD-ASO and TrAdaBoost in the following way. As for SVD-ASO, we made 400 auxiliary problems from the labels of each corpus by splitting features randomly, and extracted 50 additional features each for 4 feature groups. In total, we made new 200 additional features from 2,000 auxiliary problems. As recommended by Ando et al. (2005), we removed negative weights, performed SVD to each feature group, and iterated ASO once. Since AdaBoost easily overfitted with our rich feature vector, we applied soft margins (Ratsch et al., 2001) to TrAdaBoost. The update parameter for source examples was calculated using the update parameter on the training data in AdaBoost and the original parameter in TrAdaBoost. This ensures that the parameter would be the same as the original parameter, when the C value in the soft margin approaches infinity.

	aSVM		SVD-ASO		TrAdaBoost		SVM-CW		L2-SVM	
	F	AUC	F	AUC	F	AUC	F	AUC	F	AUC
AIMed	63.6	88.4	62.9	88.3	63.4	88.4	**64.0**	**88.6**	63.2	88.4
BioInfer	66.5	85.2	65.7	85.1	66.1	85.2	**66.7**	**85.4**	66.2	85.1
HPRD50	71.2	84.3	68.7	80.8	72.6	85.3	**72.7**	**86.4**	67.2	80.7
IEPA	73.8	85.4	72.3	83.8	74.3	**86.3**	**75.2**	85.9	73.0	84.7
LLL	85.9	89.2	79.3	85.5	86.5	88.8	**86.9**	**90.3**	80.3	86.3

Table 3: Comparison of methods on multiple corpora. Our feature vector without graph features is used. The source corpora with the best F-scores are reported for aSVM, TrAdaBoost, and SVM-CW.

	F-score						AUC					
	A	B	H	I	L	all	A	B	H	I	L	all
A	(64.2)	64.0	64.7	**65.2**	63.7	64.2	(89.1)	**89.5**	89.2	89.3	89.0	89.4
B	67.9	(67.6)	67.9	67.9	67.7	**68.3**	86.2	(86.1)	86.2	86.3	86.2	**86.4**
H	71.3	71.2	(69.7)	74.1	70.8	**74.9**	84.7	85.0	(82.8)	85.0	83.4	**87.9**
I	74.4	75.6	73.7	(74.4)	74.4	**76.6**	86.7	87.1	85.4	(85.6)	86.9	**87.8**
L	83.2	85.9	82.0	**86.7**	(80.5)	84.1	86.3	87.1	87.4	**90.8**	(86.0)	86.2

Table 4: F-score and AUC by SVM-CW. Rows correspond to a target corpus, and columns a source corpus. A:AIMed, B:BioInfer, H:HPRD50, I:IEPA, and L:LLL corpora. "all" signifies that all source corpora are used as one source corpus, ignoring the differences among the corpora. For the comparison, we show the 10-fold CV result on each target corpus.

In Table 3, we demonstrate the results of the comparison. SVM-CW improved the classification performance at least as much as all the other methods. The improvement is mainly attributed to the aggressive use of source examples while learning the model. Some source examples can be used as training data, as indicated in Figure 8. SVM-CW does not set the restriction between C_s and C_t in Equation (2), so it can use source examples aggressively while learning the model. Since aSVM transfers a model, and SVD-ASO transfers an additional feature space, aSVM and SVD-ASO do not use the source examples while learning the model. In addition to the difference in the data usage, the settings of aSVM and SVD-ASO do not match the current task. As for aSVM, the DA assumption (that the labels are the same) does not match the task. In SVD-ASO, the numbers of both source examples and auxiliary problems are much smaller than those reported by Ando et al. (2005). TrAdaBoost uses the source examples while learning the model, but never increases the weight of the examples, and it attempts to reduce their effects.

Finally, we apply SVM-CW to all corpora using all features. Table 4 summarizes the F-score and AUC by SVM-CW with all features. SVM-CW is especially effective for small corpora, showing that SVM-CW can adapt source corpora to a small annotated target corpus. The improvement on AIMed is small compared to the improvement on BioInfer, even though these corpora are similar in size. One of the reasons for this is that whole abstracts are annotated in AIMed, therefore making the examples biased. The difference between L2-SVM and SVM-CW + IEPA on AIMed is small, but statistically, it is significant (McNemar test (McNemar, 1947), P = 0.0081). In the cases of HPRD50 + IEPA, LLL + IEPA, and two folds in BioInfer + IEPA, C_s is larger than C_t in Equation (2). This is worth noting, because the source corpus is more weighted than the target corpus, and the prediction performance on the target corpus is improved. Most methods put more trust in the target corpus than in the source corpus, and our results show that this setting is not always effective for mixing corpora. The results also indicate that IEPA contains more useful information for extracting PPI than other corpora, and that using source examples aggressively is important for these combinations. We compared the results of L2-SVM and SVM-CW + IEPA on AIMed, and found that 38 pairs were described as "interaction" or "binding" in the sentences among 61

	SVM-CW		L2-SVM		Airola et al.	
	F	AUC	F	AUC	F	AUC
A	**65.2**	**89.3**	64.2	89.1	56.4	84.8
B	**68.3**	**86.4**	67.6	86.1	61.3	81.9
H	**74.9**	**87.9**	69.7	82.8	63.4	79.7
I	**76.6**	**87.8**	74.4	85.6	75.1	85.1
L	**86.7**	**90.8**	80.5	86.0	76.8	83.4

Table 6: Comparison with the results by Airola et al. (2008). A:AIMed, B:BioInfer, H:HPRD50, I:IEPA, and L:LLL corpora. The results with the highest F-score from Table 4 are reported as the results for SVM-CW.

newly found pairs. This analysis is evidence that IEPA contains instances to help find such interactions, and that SVM-CW helps to collect gold pairs that lack enough supporting instances in a single corpus, by adding instances from other corpora. SVM-CW missed coreferential relations that were also missed by L2-SVM. This can be attributed to the fact that the coreferential information is not stored in our current feature vector; so we need an even more expressive feature space. This is left as future work.

SVM-CW is effective on most corpus combinations, and all the models from single corpora can be improved by adding other source corpora. This result is impressive, because the baselines by L2-SVM on just single corpora are already better than or at least comparable to other state-of-the-art PPI extraction systems, and also because the variety of the differences among different corpora is quite wide depending on various factors including annotation policies of the corpora (Pyysalo et al., 2008). The results suggest that SVM-CW is useful as an ITL method.

4.4 Comparison with Other PPI Systems

We compare our system with other previously published PPI extraction systems. Tables 5 and 6 summarize the comparison. Table 5 summarizes the comparison of several PPI extraction systems evaluated on the AIMed corpus. As indicated, the performance of the heavy kernel method is lower than our fast rich feature-vector method. Our system is, to the extent of our knowledge, the best performing PPI extraction system evaluated on the AIMed corpus, both in terms of AUC and F-scores. Airola et al. (2008) first reported results using all five corpora. We cannot directly compare our result with the F-score results, because they tuned the threshold, but our system still outperforms the system by Airola et al. (2008) on every corpus in AUC values. The results also indicate that our system outperforms other systems on all PPI corpora, and that both the rich feature vector and the corpus weighting are effective for the PPI extraction task.

5 Conclusion

In this paper, we proposed a PPI extraction system with a rich feature vector, using a corpus weighting method (SVM-CW) for combining the multiple PPI corpora. The feature vector extracts as much information as possible from the main training corpus, and SVM-CW incorporate other external source corpora in order to improve the performance of the classifier on the main target corpus. To the extent of our knowledge, this is the first application of ITL and DA methods to PPI extraction. As a result, the system, with SVM-CW and the feature vector, outperformed all other PPI extraction systems on all of the corpora. The PPI corpora share some information, and it is shown to be effective to add other source corpora when working with a specific target corpus.

The main contributions of this paper are: 1) conducting experiments in extracting PPI using multiple corpora, 2) suggesting a rich feature vector using several previously proposed features and normalization methods, 3) the combination of SVM with corpus weighting and the new feature vector improved results on this task compared with prior work.

There are many differences among the corpora that we used, and some of the differences are still unresolved. For further improvement, it would be necessary to investigate what is shared and what is different among the corpora. The SVM-CW method, and the PPI extraction system, can be applied generally to other classification tasks, and to other binary relation extraction tasks, without the need for modification. There are several other tasks in which many different corpora, which at first glance seem compatible, exist. By applying SVM-CW to such corpora, we will analyze which differences can be resolved by SVM-CW, and what differences require a manual resolution.

For the PPI extraction system, we found many false negatives that need to be resolved. For further improvement, we need to analyze the cause

	positive	all	P	R	F	AUC
SVM-CW	1,000	5,834	60.0	**71.9**	**65.2**	**89.3**
L2-SVM	1,000	5,834	62.7	66.6	64.2	89.1
(Miwa et al., 2008)	1,005	5,648	60.4	69.3	64.2 (61.5)	87.9 (87.1)
(Miyao et al., 2008)	1,059	5,648	54.9	65.5	59.5	
(Airola et al., 2008)	1,000	5,834	52.9	61.8	56.4	84.8
(Sætre et al., 2007)	1,068	5,631	64.3	44.1	52.0	
(Erkan et al., 2007)	951	4,020	59.6	60.7	60.0	
(Bunescu and Mooney, 2005)			**65.0**	46.4	54.2	

Table 5: Comparison with previous PPI extraction results on the AIMed corpus. The numbers of positive and all examples, precision (P), recall (R), F-score (F), and AUC are shown. The result with the highest F-score from Table 4 is reported as the result for SVM-CW. The scores in the parentheses of Miwa et al. (2008) indicate the result using the same 10-fold splits as our result, as indicated in Section 4.2.

of these false negatives more deeply, and design a more discriminative feature space. This is left as a future direction of our work.

Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan), Genome Network Project (MEXT, Japan), and Scientific Research (C) (General) (MEXT, Japan).

References

Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross corpus learning. *BMC Bioinformatics*.

Rie Kubota Ando, Tong Zhang, and Peter Bartlett. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Rie Kubota Ando. 2006. Applying alternating structure optimization to word sense disambiguation. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 77–84, June.

C. M. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press.

Razvan C. Bunescu and Raymond J. Mooney. 2005. Subsequence kernels for relation extraction. In *NIPS 2005*.

Razvan C. Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun K. Ramani, and Yuk Wah Wong. 2005. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2):139–155.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP 2002*, pages 1–8.

Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. 2007. Boosting for transfer learning. In *ICML 2007*, pages 193–200.

J. Ding, D. Berleant, D. Nettleton, and E. Wurtele. 2002. Mining medline: abstracts, sentences, or phrases? *Pacific Symposium on Biocomputing*, pages 326–337.

Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. In *ICML 2008*, pages 264–271.

Gunes Erkan, Arzucan Ozgur, and Dragomir R. Radev. 2007. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In *EMNLP 2007*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2006. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. 2008. A dual coordinate descent method for large-scale linear SVM. In *ICML 2008*, pages 408–415.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus — a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19:i180–i182.

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008a. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9:10.

Seonho Kim, Juntae Yoon, and Jihoon Yang. 2008b. Kernel approaches for genic interaction extraction. *Bioinformatics*, 24(1):118–126.

Suresh Mathivanan, Balamurugan Periaswamy, TKB Gandhi, Kumaran Kandasamy, Shubha Suresh, Riaz Mohmood, YL Ramachandra, and Akhilesh Pandey. 2006. An evaluation of human protein-protein interaction data in the public domain. *BMC Bioinformatics*, 7 Suppl 5:S19.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, June.

Makoto Miwa, Rune Sætre, Yusuke Miyao, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Combining multiple layers of syntactic information for protein-protein interaction extraction. In *Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM 2008)*, pages 101–108.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics (ACL'08:HLT)*.

Claire Nédellec. 2005. Learning language in logic - genic interaction extraction challenge. In *Proceedings of the LLL'05 Workshop*.

Sinno Jialin Pan and Qiang Yang. 2008. A survey on transfer learning. Technical Report HKUST-CS08-08, Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China, November.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. 2007. BioInfer: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8:50.

Sampo Pyysalo, Antti Airola, Juho Heimonen, Jari Björne, Filip Ginter, and Tapio Salakoski. 2008. Comparative analysis of five protein-protein interaction corpora. In *BMC Bioinformatics*, volume 9(Suppl 3), page S6.

Gunnar Ratsch, Takashi Onoda, and Klaus-Robert Muller. 2001. Soft margins for adaboost. *Machine Learning*, 42(3):287–320.

Rune Sætre, Kenji Sagae, and Jun'ichi Tsujii. 2007. Syntactic features for protein-protein interaction extraction. In *LBM 2007 short papers*.

Gabriele Schweikert, Christian Widmer, Bernhard Schölkopf, and Gunnar Rätsch. 2008. An empirical analysis of domain adaptation algorithms for genomic sequence analysis. In *NIPS*, pages 1433–1440.

Jun Yang, Rong Yan, and Alexander G. Hauptmann. 2007. Cross-domain video concept detection using adaptive SVMs. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 188–197.

Generalized Expectation Criteria for Bootstrapping Extractors using Record-Text Alignment

Kedar Bellare
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
kedarb@cs.umass.edu

Andrew McCallum
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
mccallum@cs.umass.edu

Abstract

Traditionally, machine learning approaches for information extraction require human annotated data that can be costly and time-consuming to produce. However, in many cases, there already exists a database (DB) with schema related to the desired output, and records related to the expected input text. We present a conditional random field (CRF) that aligns tokens of a given DB record and its realization in text. The CRF model is trained using only the available DB and unlabeled text with generalized expectation criteria. An annotation of the text induced from inferred alignments is used to train an information extractor. We evaluate our method on a citation extraction task in which alignments between DBLP database records and citation texts are used to train an extractor. Experimental results demonstrate an error reduction of 35% over a previous state-of-the-art method that uses heuristic alignments.

1 Introduction

A substantial portion of information on the Web consists of unstructured and semi-structured text. Information extraction (IE) systems segment and label such text to populate a structured database that can then be queried and mined efficiently.

In this paper, we mainly deal with information extraction from text fragments that closely resemble structured records. Examples of such texts include citation strings in research papers, contact addresses on person homepages and apartment listings in classified ads. Pattern matching and rule-based approaches for IE (Brin, 1998; Agichtein and Gravano, 2000; Etzioni et al., 2005) that only use specific patterns, and delimiter and font-based cues for segmentation are prone to failure on such data because these cues are generally not broadly reliable. Statistical machine learning methods such as hidden Markov models (HMMs) (Rabiner, 1989; Seymore et al., 1999; Freitag and McCallum, 1999) and conditional random fields (CRFs) (Lafferty et al., 2001; Peng and McCallum, 2004; Sarawagi and Cohen, 2005) have become popular approaches to address the text extraction problem. However, these methods require labeled training data, such as annotated text, which is often scarce and expensive to produce.

In many cases, however, there already exists a database with schema related to the desired output, and records that are imperfectly rendered in the available unlabeled text. This database can serve as a source of significant supervised guidance to machine learning methods. Previous work on using databases to train information extractors has taken one of three simpler approaches. In the first, a separate language model is trained on each column of the database and these models are then used to segment and label a given text sequence (Agichtein and Ganti, 2004; Canisius and Sporleder, 2007). However, this approach does not model context, errors or different formats of fields in text, and requires large number of database entries to learn an accurate language model. The second approach (Sarawagi and Cohen, 2004; Michelson and Knoblock, 2005; Mansuri and Sarawagi, 2006) uses database or dictionary lookups in combination with similarity measures to add features to the text sequence. Although these features are very informative, learning algorithms still require annotated data to make use of them. The final approach heuristically labels texts using matching records and learns extractors from these annotations (Ramakrishnan and Mukherjee, 2004; Bellare and McCallum, 2007; Michelson and Knoblock, 2008). Heuris-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 131–140,
Singapore, 6-7 August 2009.

tic labeling decisions, however, are made independently without regard for the Markov dependencies among labels in text and are sensitive to subtle changes in text.

Here we propose a method that automatically induces a labeling of an input text sequence using a word *alignment* with a matching database record. This induced labeling is then used to train a text extractor. Our approach has several advantages over previous methods. First, we are able to model field ordering and context around fields by learning an extractor from annotations of the text itself. Second, a probabilistic model for word alignment can exploit dependencies among alignments, and is also robust to errors, formatting differences, and missing fields in text and the record.

Our word alignment model is a conditional random field (CRF) (Lafferty et al., 2001) that generates alignments between tokens of a text sequence and a matching database record. The structure of the graphical model resembles IBM Model 1 (Brown et al., 1993) in which each target (record) word is assigned one or more source (text) words. The alignment is generated conditioned on both the record and text sequence, and therefore supports large sets of rich and non-independent features of the sequence pairs. Our model is trained without the need for labeled word alignments by using generalized expectation (GE) criteria (Mann and McCallum, 2008) that penalize the divergence of specific model expectations from target expectations. Model parameters are estimated by minimizing this divergence. To limit over-fitting we include a $\mathbb{L}_2$-regularization term in the objective. The model expectations in GE criteria are taken with respect to a set of alignment latent variables that are either specific to each sequence pair (local) or summarizing the entire data set (global). This set is constructed by including all alignment variables a that satisfy a certain binary feature (e.g., $f(a, \mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2) = 1$, for labeled record $(\mathbf{x}_1, \mathbf{y}_1)$, and text sequence $\mathbf{x}_2$). One example global criterion is that "an alignment exists between two orthographically similar[1] words 95% of the time." Here the criterion has a *target expectation* of 95% and is defined over alignments $\{a = \langle i, j \rangle \mid \mathbf{x}_1[i] \sim \mathbf{x}_2[j], \forall \mathbf{x}_1, \mathbf{x}_2\}$. Another criterion for extraction can be "the word 'EMNLP' is always aligned with the record label *booktitle*". This criterion has a *target* of 100% and defined for $\{a = \langle i, j \rangle \mid \mathbf{y}_1[i] = \textit{booktitle} \wedge \mathbf{x}_2[j] = \text{'EMNLP'}, \forall \mathbf{y}_1, \mathbf{x}_2\}$. One-to-one correspondence between words in the sequence pair can be specified as collection of local expectation constraints. Since we directly encode prior knowledge of how alignments behave in our criteria, we obtain sufficiently accurate alignments with little supervision.

We apply our method to the task of citation extraction. The input to our training algorithm is a set of matching DBLP[2]-record/citation-text pairs and global GE criteria[3] of the following two types: (1) *alignment* criteria that consider features of mapping between record and text words, and, (2) *extraction* criteria that consider features of the schema label assigned to a text word. In our experiments, the parallel record-text pairs are collected manually but this process can be automated using systems that match text sequences to records in the DB (Michelson and Knoblock, 2005; Michelson and Knoblock, 2008). Such systems achieve very high accuracy close to 90% F1 on semi-structured domains similar to ours.[4] Our trained alignment model can be used to directly align new record-text pairs to create a labeling of the texts. Empirical results demonstrate a 20.6% error reduction in token labeling accuracy compared to a strong baseline method that employs a set of high-precision alignments. Furthermore, we provide a 63.8% error reduction compared to IBM Model 4 (Brown et al., 1993). Alignments learned by our model are used to train a linear-chain CRF extractor. We obtain an error reduction of 35.1% over a previous state-of-the-art extraction method that uses heuristically generated alignments.

2 Record-Text Alignment

Here we provide a brief description of the record-text alignment task. For the sake of clarity and space, we describe our approach on a *fictional* restaurant address data set. The input to our system is a database (DB) consisting of records (possibly containing errors) and corresponding texts that are realizations of these DB records. An example of a matching record-text pair is shown in

[1] Two words are orthographically similar if they have low edit distance.

[2] http://www.informatik.uni-trier.de/~ley/db/

[3] Expectation criteria used in our experiments are listed at *http://www.cs.umass.edu/~kedarb/dbie_expts.txt*.

[4] To obtain more accurate record-text pairs, matching methods can be tuned for high precision at the expense of recall. Alternatively, humans can cheaply provide match/mismatch labels on automatically matched pairs.

Record

name	address	city	state	phone
restaurant katsu	*n. hillhurst avenue*	*los angeles*		*665-1891*

Text

katzu, 1972 hillhurst ave., los feliz, california

Table 1: An example of a matching record-text pair for restaurant addresses.

Table 1. This example displays the differences between the record and text: (1) spelling errors: *katsu* → *katzu*, (2) word insertions (*restaurant*), deletions (*1972*), substitutions (*angeles* → *feliz*), (3) abbreviations (*avenue* → *ave.*), (4) missing fields in text (**phone**=*665-1891*), and (5) extra fields in text (**state**=*california*). These discrepancies plus the unknown ordering of fields within text can be addressed through word alignment.

	katzu,	*1972*	*hillhurst*	*ave.,*	*los*	*feliz,*	*california*
restaurant **[name]**	.	.	.	.	.	.	.
katsu **[name]**	■	.	.	.	.	.	.
null **[name]**	.	.	.	.	.	.	.
n. **[address]**	.	.	.	.	.	.	.
hillhurst **[address]**	.	.	■	.	.	.	.
avenue **[address]**	.	.	.	■	.	.	.
null **[address]**	.	■	.	.	.	.	.
los **[city]**	.	.	.	.	■	.	.
angeles **[city]**	.	.	.	.	.	■	.
null **[city]**	.	.	.	.	.	.	.
null **[state]**	.	.	.	.	.	.	■
665-1891 **[phone]**	.	.	.	.	.	.	.
null **[phone]**	.	.	.	.	.	.	.

Table 2: Example of a word alignment. ■ represents aligned tokens. Vertical text at the bottom are the text tokens. Horizontal text on the left are tokens from the DB record with labels shown in braces.

An example word alignment between the record and text is shown in Table 2. Tokenization of record/text string is based on whitespace characters. We add a special *null* token at the field boundaries for each label in the schema to model word insertions. The record sequence is obtained by concatenating individual fields according to the DB schema order. As in statistical word alignment, we assume the DB record to be our source and the text to be our target. The induced labeling of the text is given by (**name**, **address**, **address**, **city**, **city**, **state**) which can be used to train an information extractor. In the next section, we present our approach to address this task.

3 Approach

We first define notation that will be used throughout this section. Let $(\mathbf{x}_1, \mathbf{y}_1)$ be a database record with token sequence $\mathbf{x}_1 = \langle x_1[1], x_1[2], \ldots, x_1[m] \rangle$ and label sequence $\mathbf{y}_1 = \langle y_1[1], y_1[2], \ldots, y_1[m] \rangle$ with $y_1[*] \in \mathcal{Y}$ where $\mathcal{Y}$ is the database schema. Let $\mathbf{x}_2 = \langle x_2[1], x_2[2], \ldots, x_2[n] \rangle$ be the text sequence. Let $\mathbf{a} = \langle a_1, a_2, \ldots, a_n \rangle$ be an alignment sequence of same length as the target text sequence. The alignment $a_i = j$ assigns the DB token-label pair $(x_1[j], y_1[j])$ to the text token $x_2[i]$.

3.1 Conditional Random Field for Alignment

Our conditional random field (CRF) for alignment has a graphical model structure that resembles that of IBM Model 1 (Brown et al., 1993). The CRF is an undirected graphical model that defines a probability distribution over alignment sequences $\mathbf{a}$ conditioned on the inputs $(\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2)$ as:

$$p_\Theta(\mathbf{a}|\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2) = \frac{\exp(\sum_{t=1}^{n} \Theta^\top \vec{f}(a_t, \mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, t))}{Z_\Theta(\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2)}, \quad (1)$$

where $\vec{f}(a_t, \mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, t)$ are feature functions defined over the alignments and inputs, Θ are the model parameters and $Z_\Theta(\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2) = \sum_{\mathbf{a}'} \exp(\sum_{t=1}^{n} \Theta^\top \vec{f}(a'_t, \mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, t))$ is the partition function.

The feature vector $\vec{f}(a_t, \mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, t)$ is the concatenation of two types of feature functions: (1) *alignment* features $f_{align}(a_t, \mathbf{x}_1, \mathbf{x}_2, t)$ defined on source-target tokens, and, (2) *extraction* features $f_{extr}(a_t, \mathbf{y}_1, \mathbf{x}_2, t)$ defined on source labels and target text. To obtain the probability of an alignment in a particular position t we marginalize out the alignments over the rest of the positions $\{1, \ldots, n\} \backslash \{t\}$,

$$p_\Theta(a_t|\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2) = \sum_{\{a[1\ldots n]\} \backslash \{a_t\}} p_\Theta(\mathbf{a}|\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2)$$

$$= \frac{\exp(\Theta^\top \vec{f}(a_t, \mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, t))}{\exp(\sum_{a'} \Theta^\top \vec{f}(a', \mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, t))} \quad (2)$$

Furthermore, the marginal over label y_t assigned to the text token $x_2[t]$ at time step t during alignment is given by

$$p_\Theta(y_t|\mathbf{x}_2) = \sum_{\{a_t|\mathbf{y}_1[a_t]=y_t\}} p_\Theta(a_t|\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2), \quad (3)$$

where $\{a_t \mid \mathbf{y}_1[a_t] = y_t\}$ is the set of alignments that result in a labeling y_t for token $x_2[t]$. Henceforth, we abbreviate $p_\Theta(a_t|\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2)$ to $p_\Theta(a_t)$. The gradient of $p_\Theta(a_t)$ with respect to parameters Θ is given by

$$\frac{\partial p_\Theta(a_t)}{\partial \Theta} = p_\Theta(a_t) \left[\vec{f}(a_t, \mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, t) - E_{p_\Theta(a)} \left(\vec{f}(a, \mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, t) \right) \right], \quad (4)$$

where the expectation term in the above equation sums over all alignments a at position t. We use the Baum-Welch and Viterbi algorithms to compute marginal probabilities and best alignment sequences respectively.

3.2 Expectation Criteria and Parameter Estimation

Let $\mathcal{D} = \langle (\mathbf{x}_1^{(1)}, \mathbf{y}_1^{(1)}, \mathbf{x}_2^{(1)}), \ldots, (\mathbf{x}_1^{(K)}, \mathbf{y}_1^{(K)}, \mathbf{x}_2^{(K)}) \rangle$ be a data set of K record-text pairs gathered manually or automatically through matching (Michelson and Knoblock, 2005; Michelson and Knoblock, 2008). A global expectation criterion is defined on the set of alignment latent variables $\mathbf{A}_f = \{a | f(a, \mathbf{x}_1^{(i)}, \mathbf{y}_1^{(i)}, \mathbf{x}_2^{(i)}) = 1, \forall i = 1 \ldots K\}$ on the entire data set that satisfy a given binary feature $f(a, \mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2)$. Similarly a local expectation criterion is defined only for a specific instance $(\mathbf{x}_1^{(i)}, \mathbf{y}_1^{(i)}, \mathbf{x}_2^{(i)})$ with the set $\mathbf{A}_f = \{a | f(a, \mathbf{x}_1^{(i)}, \mathbf{y}_1^{(i)}, \mathbf{x}_2^{(i)}) = 1\}$. For a feature function f, a target expectation p, and, a weight w, our criterion minimizes the squared divergence

$$\Delta(f, p, w, \Theta) = w \left(\frac{E_{p_\Theta}(\mathbf{A}_f)}{|\mathbf{A}_f|} - p \right)^2, \quad (5)$$

where $E_{p_\Theta}(\mathbf{A}_f) = \sum_{a \in \mathbf{A}_f} p_\Theta(a)$ is the sum of marginal probabilities given by Equation (2) and $|\mathbf{A}_f|$ is the size of the variable set. The weight w influences the importance of satisfying a given expectation criterion. Equation (5) is an instance of generalized expectation criteria (Mann and McCallum, 2008) that penalizes the divergence of a specific model expectation from a given target value. The gradient of the divergence with respect to Θ is given by,

$$\frac{\partial \Delta(f, p, w, \Theta)}{\partial \Theta} = 2w \left(\frac{E_{p_\Theta}(\mathbf{A}_f)}{|\mathbf{A}_f|} - p \right) \times \left[\frac{1}{|\mathbf{A}_f|} \sum_{a \in \mathbf{A}_f} \frac{\partial p_\Theta(a)}{\partial \Theta} - p \right], \quad (6)$$

where the gradient $\frac{\partial p_\Theta(a)}{\partial \Theta}$ is given by Eq. (4). Given expectation criteria $\mathcal{C} = \langle \mathbf{F}, \mathbf{P}, \mathbf{W} \rangle$ with a set of binary feature functions $\mathbf{F} = \langle f_1, \ldots, f_l \rangle$, target expectations $\mathbf{P} = \langle p_1, \ldots, p_l \rangle$ and weights $\mathbf{W} = \langle w_1, \ldots, w_l \rangle$, we maximize the objective

$$\mathcal{O}(\theta; \mathcal{D}, \mathcal{C}) = \max_\Theta - \sum_{i=1}^{l} \Delta(f_i, p_i, w_i, \Theta) - \frac{||\Theta||^2}{2}, \quad (7)$$

where $||\Theta||^2/2$ is the regularization term added to limit over-fitting. Hence the gradient of the objective is

$$\frac{\partial \mathcal{O}(\theta; \mathcal{D}, \mathcal{C})}{\partial \Theta} = - \sum_{i=1}^{l} \frac{\partial \Delta(f_i, p_i, w_i, \Theta)}{\partial \Theta} - \Theta.$$

We maximize our objective (Equation 7) using the L-BFGS algorithm. It is sometimes necessary to restart maximization after resetting the Hessian calculation in L-BFGS due to non-convexity of our objective.[5] Also, non-convexity may lead to a local instead of a global maximum. Our experiments show that local maxima do not adversely affect performance since our accuracy is within 4% of a model trained with gold-standard labels.

3.3 Linear-chain CRF for Extraction

The alignment CRF (**AlignCRF**) model described in Section 3.1 is able to predict labels for a text sequence given a matching DB record. However, without corresponding records for texts the model does not perform well as an extractor because it has learned to rely on the DB record and alignment features (Sutton et al., 2006). Hence, we train a separate linear-chain CRF on the alignment-induced labels for evaluation as an extractor.

The extraction CRF (**ExtrCRF**) employs a fully-connected state machine with a unique state

[5] Our L-BFGS optimization procedure checks whether the approximate Hessian computed from cached gradient vectors is positive semi-definite. The optimization is restarted if this check fails.

per label $y \in \mathcal{Y}$ in the database schema. The CRF induces a conditional probability distribution over label sequences $\mathbf{y} = \langle y_1, \ldots, y_n \rangle$ and input text sequences $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ as

$$p_\Lambda(\mathbf{y}|\mathbf{x}) = \frac{\exp\left(\sum_{t=1}^{n} \Lambda^\top \vec{g}(y_{t-1}, y_t, \mathbf{x}, t)\right)}{Z_\Lambda(\mathbf{x})}. \quad (8)$$

In comparison to our earlier zero-order **AlignCRF** model, our **ExtrCRF** is a first-order model. All the feature functions in this model $g(y_{t-1}, y_t, \mathbf{x}, t)$ are a conjunction of the label pair (y_{t-1}, y_t) and input observational features. $Z_\Lambda(\mathbf{x})$ in the equation above is the partition function. Inference in the model is performed using the Viterbi algorithm.

Given expectation criteria $\mathcal{C}$ and data set $\mathcal{D} = \langle (\mathbf{x}_1^{(1)}, \mathbf{y}_1^{(1)}, \mathbf{x}_2^{(1)}), \ldots, (\mathbf{x}_1^{(K)}, \mathbf{y}_1^{(K)}, \mathbf{x}_2^{(K)}) \rangle$, we first estimate the parameters Θ of **AlignCRF** model as described in Section 3.2. Next, for all text sequences $\mathbf{x}_2^{(i)}, i = 1 \ldots K$ we compute the marginal probabilities of the labels $p_\Theta(y_t|\mathbf{x}_2^{(i)}), \forall t$ using Equation (3). To estimate parameters Λ we minimize the KL-divergence between $p_\Theta(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{n} p_\Theta(y_t|\mathbf{x})$ and $p_\Lambda(\mathbf{y}|\mathbf{x})$ for all sequences $\mathbf{x}$,

$$\begin{aligned} KL(p_\Theta \| p_\Lambda) &= \sum_{\mathbf{y}} p_\Theta(\mathbf{y}|\mathbf{x}) \log\left(\frac{p_\Theta(\mathbf{y}|\mathbf{x})}{p_\Lambda(\mathbf{y}|\mathbf{x})}\right) \\ &= H(p_\Theta(\mathbf{y}|\mathbf{x})) \\ &- \sum_{t, y_{t-1}, y_t} E_{p_\Theta(y_{t-1}, y_t)}[\Lambda^\top \vec{g}(y_{t-1}, y_t, \mathbf{x}, t)] \\ &+ \log(Z_\Lambda(\mathbf{x})). \quad (9) \end{aligned}$$

The gradient of the above equation is given by

$$\begin{aligned} \frac{\partial KL}{\partial \Lambda} &= \sum_{t, y_{t-1}, y_t} E_{p_\Lambda(y_{t-1}, y_t|\mathbf{x})}[\vec{g}(y_{t-1}, y_t, \mathbf{x}, t)] \\ &- E_{p_\Theta(y_{t-1}, y_t|\mathbf{x})}[\vec{g}(y_{t-1}, y_t, \mathbf{x}, t)]. \quad (10) \end{aligned}$$

Both the expectations can be computed using the Baum-Welch algorithm. The parameters Λ are estimated for a given data set $\mathcal{D}$ and learned parameters Θ by optimizing the objective

$$\begin{aligned} O(\Lambda; \mathcal{D}, \Theta) = \min_\Lambda \sum_{i=1}^{K} KL(p_\Theta(\mathbf{y}|\mathbf{x}_2^{(i)}) \| p_\Lambda(\mathbf{y}|\mathbf{x}_2^{(i)}) \\ + \|\Lambda\|^2/2. \end{aligned}$$

The objective is minimized using L-BFGS. Since the objective is convex we are guaranteed to obtain a global minima.

4 Experiments

In this section, we present details about the application of our method to citation extraction task.

Data set. We collected a set of 260 random records from the DBLP bibliographic database. The schema of DBLP has the following labels {*author, editor, address, title, booktitle, pages, year, journal, volume, number, month, url, ee, cdrom, school, publisher, note, isbn, chapter, series*}. The complexity of our alignment model depends on the number of schema labels and number of tokens in the DB record. We reduced the number of schema labels by: (1) mapping the labels *address, booktitle, journal* and *school* to *venue*, (2) mapping *month* and *year* to *date*, and (3) dropping the fields *url, ee, cdrom, note, isbn* and *chapter*, since they never appeared in citation texts. We also added the other label *O* for fields in text that are not represented in the database. Therefore, our final DB schema is {*author, title, date, venue, volume, number, pages, editor, publisher, series, O*}.

For each DBLP record we searched on the web for matching citation texts using the first author's last name and words in the title. Each citation text found is manually labeled for evaluation purposes. An example of a matching DBLP record-citation text pair is shown in Table 3. Our data set[6] contains 522 record-text pairs for 260 DBLP entries.

Features and Constraints. We use a variety of rich, non-independent features in our models to optimize system performance. The input features in our models are of the following two types:

(a) Extraction features in the **AlignCRF** model ($f(a_t, \mathbf{y}_1, \mathbf{x}_2, t)$) and **ExtrCRF** model ($g(y_{t-1}, y_t, \mathbf{x}, t)$) are conjunctions of assigned labels and observational tests on text sequence at time step t. The following observational tests are used: (1) regular expressions to detect tokens containing all characters (ALLCHAR), all digits (ALLDIGITS) or both digits and characters (ALPHADIGITS), (2) number of characters or digits in the token (NUMCHAR=3, NUMDIGITS=1), (3) domain-specific patterns for *date* and *pages*, (4) token identity, suffixes, prefixes and character n-grams, (5) presence of a token in lexicons such as "last names," "publisher names," "cities," (6) lexicon features within a window of 10, (7) regular

[6]The data set can be found at *http://www.cs.umass.edu/∼kedarb/dbie_cite_data.sgml*.

DBLP record	Citation text
[Chengzhi Li]$_{author}$ [Edward W. Knightly]$_{author}$ [Coordinated Network Scheduling: A Framework for End-to-End Services.]$_{title}$ [69-]$_{pages}$ [2000]$_{date}$ [ICNP]$_{venue}$	[C. Li]$_{author}$ [and]$_{O}$ [E. Knightly.]$_{author}$ [Coordinated network scheduling: A framework for end-to-end services.]$_{title}$ [In Proceedings of IEEE ICNP]$_{venue}$ ['00,]$_{date}$ [Osaka, Japan,]$_{venue}$ [November 2000.]$_{date}$

Table 3: Example of matching record-text pair found on the web.

expression features within a window of 10, and (8) token identity features within a window of 3.

(b) Alignment features in the **AlignCRF** model ($f(a_t, \mathbf{x}_1, \mathbf{x}_2, t)$) that operate on the aligned source token $\mathbf{x}_1[a_t]$ and target token $\mathbf{x}_2[t]$. Again the observational tests used for alignment are: (1) exact token match tests whether the source-target tokens are string identical, (2) approximate token match produces a binary feature after binning the Jaro-Winkler edit distance (Cohen et al., 2003) between the tokens, (3) substring token match tests whether one token is a substring of the other, (4) prefix token match returns true if the prefixes match for lengths $\{1, 2, 3, 4\}$, (5) suffix token match returns true if the prefixes match for lengths $\{1, 2, 3, 4\}$, and (6) exact and approximate token matches at offsets $\{-1, -1\}$ and $\{+1, +1\}$ around the alignment.

Thus, a conditional model lets us use these arbitrary helpful features that cannot be exploited tractably in a generative model.

As is common practice (Haghighi and Klein, 2006; Mann and McCallum, 2008), we simulate user-specified expectation criteria through statistics on manually labeled citation texts. For extraction criteria, we select for each label, the top N extraction features ordered by mutual information (MI) with that label. Also, we aggregate the alignment features of record tokens whose alignment with a target text token results in a correct label assignment. The top N alignment features that have maximum MI with this correct labeling are selected as alignment criteria. We bin target expectations of these criteria into 11 bins as $[0.05, 0.1, 0.2, 0.3, \ldots, 0.9, 0.95]$.[7] In our experiments, we set $N = 10$ and use a fixed weight $w = 10.0$ for all expectation criteria (no tuning of parameters was performed). Table 4 shows a sample of GE criteria used in our experiments.[8]

[7] Mann and McCallum (2008) note that GE criteria are robust to deviation of specified targets from actual expectations.

[8] A complete list of expectation criteria is available at *http://www.cs.umass.edu/∼kedarb/dbie_expts.txt*.

Label	Feature	Prior
alignment	PREFIX_MATCH4	0.95
author	LEXICON_LASTNAME	0.6
title	WINDOW_WORD=Maintenance	0.95
venue	WINDOW_WORD=Conference	0.95
date	YEAR_PATTERN	0.95
volume	NUMDIGITS=2	0.6
number	NUMDIGITS=1	0.6
pages	PAGES_PATTERN	0.95
editor	WORD_PREFIX[2]=ed	0.95
publisher	WORD=Press	0.95
series	WORD=Notes	0.95
O	WORD=and	0.7

Table 4: Sample of expectation criteria used by our model.

Experimental Setup. Our experiments use a 3:1 split of the data for training and testing. We repeat the experiment 20 times with different random splits of the data. We train the **AlignCRF** model using the training data and the automatically created expectation criteria (Section 3.2). We evaluate our alignment model indirectly in terms of token labeling accuracy (i.e., percentage of correctly labeled tokens in test citation data) since we do not have annotated alignments. The alignment model is then used to train a **ExtrCRF** model as described in Section 3.3. Again, we use token labeling accuracy for evaluation. We also measure F1 performance as the harmonic mean of precision and recall for each label.

4.1 Alternate approaches

We compare our method against alternate approaches that either learn alignment or extraction models from training data.

Alignment approaches. We use GIZA++ (Och and Ney, 2003) to train generative directed alignment models: **HMM** and IBM **Model4** (Brown et al., 1993) from training record-text pairs. These models are currently being used in state-of-the-art machine translation systems. Alignments between matching DB records and text sequences are then used for labeling at test time.

Extraction approaches. The first alternative (**DB-CRF**) trains a linear-chain CRF for extraction on fields of the database entries only. Each field of the record is treated as a separate labeled text sequence. Given an unlabeled text sequence, it is segmented and labeled using the Viterbi algorithm. This method is an enhanced representative for (Agichtein and Ganti, 2004) in which a language model is trained for each column of the DB.

Another alternative technique constructs partially annotated text data using the matching records and a labeling function. The labeling function employs high-precision alignment rules to assign labels to text tokens using labeled record tokens. We use exact and approximate token matching rules to create a partially labeled sequence, skipping tokens that cannot be unambiguously labeled. In our experiments, we achieve a precision of 97% and a recall of 70% using these rules. Given a partially annotated citation text, we train a linear-chain CRF by maximizing the marginal likelihood of the observed labels. This marginal CRF training method (Bellare and McCallum, 2007) (**M-CRF**) was the previous state-of-the-art on this data set. Additionally, if a matching record is available for a test citation text, we can partially label tokens and use constrained Viterbi decoding with labeled positions fixed at their observed values (**M+R-CRF** approach).

Our third approach is similar to (Mann and McCallum, 2008). We create extraction expectation criteria from labeled text sequences in the training data and uses these criteria to learn a linear-chain CRF for extraction (**MM08**). The performance achieved by this approach is an upper bound on methods that: (1) use labeled training records to create extraction criteria, and, (2) only use extraction criteria without any alignment criteria.

Finally, we train a supervised linear-chain CRF (**GS-CRF**) using the labeled text sequences from the training set. This represents an upper bound on the performance that can be achieved on our task. All the extraction methods have access to the same features as the **ExtrCRF** model.

4.2 Results

Table 5 shows the results of various alignment algorithms applied to the record-text data set. Alignment methods use the matching record to perform labeling of a test citation text. The **AlignCRF** model outperforms the best generative alignment model **Model4** (IBM Model 4) with an error reduction of 63.8%. Our conjecture is that **Model4** is getting stuck in sub-optimal local maxima during EM training since our training set only contains hundreds of parallel record-text pairs. This problem may be alleviated by training on a large parallel corpus. Additionally, our alignment model is superior to **Model4** since it leverages rich non-independent features of input sequence pairs.

	HMM	**Model4**	**AlignCRF**
accuracy	78.5%	79.8%	**92.7%**
author	92.7	94.9	**99.0**
title	93.3	95.1	**97.3**
date	69.5	66.3	**81.9**
venue	73.3	73.1	**91.2**
volume	50.0	49.2	**78.5**
number	53.5	66.3	**68.0**
pages	38.2	44.1	**88.2**
editor	22.8	21.5	**78.1**
publisher	29.7	31.0	**72.6**
series	**77.4**	77.3	74.6
O	49.6	58.8	**85.7**

Table 5: Token-labeling accuracy and per-label F1 for different alignment methods. These methods all use matching DB records at test time. Bold-faced numbers indicate the best performing model. **HMM, Model4**: generative alignment models from GIZA++, **AlignCRF**: alignment model from this paper.

Table 6 shows the performance of various extraction methods. Except **M+R-CRF**, all extraction approaches, do not use any record information at test time. In comparison to the previous state-of-the-art **M-CRF**, the **ExtrCRF** method provides an error reduction of 35.1%. **ExtrCRF** also produces an error reduction of 21.7% compared to **M+R-CRF** without the use of matching records. These reductions are significant at level $p = 0.005$ using the two-tailed t-test. Training only on DB records is not helpful for extraction as we do not learn the transition structure[9] and additional context information[10] in text. This explains the low accuracy of the **DB-CRF** method. Furthermore, the **MM08** approach (Mann and McCallum, 2008) achieves low accuracy since it does not use any

[9] In general, the *editor* field follows the *title* field while the *author* field precedes it.

[10] The token "Vol." generally precedes the *volume* field in text. Similarly, tokens "pp" and "pages" occur before the *pages* field.

	DB-CRF	M-CRF	M+R-CRF†	MM08	ExtrCRF	GS-CRF
accuracy	70.4%	88.9%	90.8%	73.5%	**92.8%**	96.5%
author	72.4	93.7	94.1	85.4	**98.5**	99.0
title	79.4	96.7	**98.4**	83.1	94.6	98.1
date	60.1	74.5	76.2	57.8	**81.7**	93.5
venue	67.3	89.4	**91.5**	73.2	89.8	95.9
volume	20.3	69.4	74.2	27.7	**78.9**	90.5
number	30.1	72.8	**80.8**	47.8	75.1	91.4
pages	41.4	80.9	84.5	49.6	**92.1**	94.1
editor	7.1	71.1	**79.3**	75.3	73.3	93.7
publisher	62.1	67.5	**77.2**	40.2	58.5	82.2
series	65.2	74.9	**76.3**	65.9	73.8	85.8
O	54.1	7.0	8.3	57.7	**91.9**	94.5

Table 6: Token-labeling accuracy and per-label F1 for different extraction methods. Except **M+R-CRF**†, all other approaches do not use any records at test time. Bold-faced numbers indicate the best performing model. **DB-CRF**: CRF trained on DB fields. **M+R-CRF**, **M-CRF**: CRFs trained from heuristic alignments. **ExtrCRF**: Extraction model presented in this paper. **GS-CRF**: CRF trained on human annotated citation texts.

alignment criteria during training. Hence, alignment information is crucial for obtaining high accuracy.

Note that we do not observe a decrease in performance of **ExtrCRF** over **AlignCRF** although we are not using the test records during decoding. This is because: (1) a first-order model in **ExtrCRF** improves performance compared to a zero-order model in **AlignCRF** and (2) the use of noisy DB records in the test set for alignment often increases extraction error.

Both our models have a high F1 value for the other label *O* because we provide our algorithm with constraints for the label *O*. In contrast, since there is no realization of the *O* field in the DB records, both **M-CRF** and **M+R-CRF** methods fail to label such tokens correctly. Our alignment model trained using expectation criteria achieves a performance of 92.7% close to gold-standard training (**GS-CRF**) (96.5%). Furthermore, **ExtrCRF** obtains an accuracy of 92.8% similar to **AlignCRF** without access to DB records due to better modeling of transition structure and context.

5 Related Work

Recent research in information extraction (IE) has focused on reducing the labeling effort needed to train supervised IE systems. For instance, Grenager et al. (2005) perform unsupervised HMM learning for field segmentation, and bias the model to prefer self-transitions and transitions on boundary tokens. Unfortunately, such unsupervised IE approaches do not attain performance close to state-of-the-art supervised methods. Semi-supervised approaches that learn a model with only a few constraints specifying prior knowledge have generated much interest. Haghighi and Klein (2006) assign each label in the model certain prototypical features and train a Markov random field for sequence tagging from these labeled features. In contrast, our method uses GE criteria (Mann and McCallum, 2008) – allowing soft-labeling of features with target expectation values – to train conditional models with complex and non-independent input features. Additionally, in comparison to previous methods, an information extractor trained from our record-text alignments achieves accuracy of 93% making it useful for real-world applications. Chang et al. (2007) use beam search for decoding unlabeled text with soft and hard constraints, and train a model with top-K decoded label sequences. However, this model requires large number of labeled examples (e.g., 300 annotated citations) to bootstrap itself. Active learning is another popular approach for reducing annotation effort. Settles and Craven (2008) provide a comparison of various active learning strategies for sequence labeling tasks. We have shown, however, that in domains where a database can provide significant supervision, one can bootstrap accurate extractors with very little human effort.

Another area of research, related to the task described in our paper, is learning extractors from database records. These records are also known as field books and reference sets in literature (Canisius and Sporleder, 2007; Michelson and Knoblock, 2008). Both Agichtein and Ganti (2004) and Canisius and Sporleder (2007) train a language model for each database column. The language modeling approach is sensitive to word re-orderings in text and other variability present in real-world text (e.g., abbreviation). We allow for word and field re-orderings through alignments and model complex transformations through feature functions. Michelson and Knoblock (2008) extract information from unstructured texts using a rule-based approach to align segments of text with fields in a DB record. Our probabilistic alignment approach is more robust and uses rich features of the alignment to obtain high performance.

Recently, Snyder and Barzilay (2007) and Liang et al. (2009) have explored record-text matching in domains with unstructured texts. Unlike a semi-structured text sequence obtained by noisily concatenating fields from a single record, an unstructured sequence may contain fields from multiple records embedded in large amounts of extraneous text. Hence, the problems of record-text matching and word alignment are significantly harder in unstructured domains. Snyder and Barzilay (2007) achieve a state-of-the-art performance of 80% F1 on matching multiple NFL database records to sentences in the news summary of a football game. Their algorithm is trained using supervised machine learning and learns alignments at the level of sentences and DB records. In contrast, this paper presents a semi-supervised learning algorithm for learning token-level alignments between records and texts. Liang et al. (2009) describe a model that simultaneously performs record-text matching and word alignment in unstructured domains. Their model is trained in an unsupervised fashion using EM. It may be possible to further improve their model performance by incorporating prior knowledge in the form of expectation criteria.

Traditionally, generative word alignment models have been trained on massive parallel corpora (Brown et al., 1993). Recently, discriminative alignment methods trained using annotated alignments on small parallel corpora have achieved superior performance. Taskar et al. (2005) train a discriminative alignment model from annotated alignments using a large-margin method. Labeled alignments are also used by Blunsom and Cohn (2006) to train a CRF word alignment model. Our method is trained using a small number of easily specified expectation criteria thus avoiding tedious and expensive human labeling of alignments. An alternate method of learning alignment models is proposed by McCallum et al. (2005) in which the training set consists of sequence pairs classified as match or mismatch. Alignments are learned to identify the class of a given sequence pair. However, this method relies on carefully selected negative examples to produce high-accuracy alignments. Our method produces good alignments as we directly encode prior knowledge about alignments.

6 Conclusion and Future Work

Information extraction is an important first step in data mining applications. Earlier approaches for learning reliable extractors have relied on manually annotated text corpora. This paper presents a novel approach for training extractors using alignments between texts and existing database records. Our approach achieves performance close to supervised training with very little supervision.

In the future, we plan to surpass supervised accuracy by applying our method to millions of parallel record-text pairs collected automatically using matching. We also want to explore the addition of Markov dependencies into our alignment model and other constraints such as monotonicity and one-to-one correspondence.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

Eugene Agichtein and Venkatesh Ganti. 2004. Mining reference tables for automatic text segmentation. In *KDD*.

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *ICDL*.

Kedar Bellare and Andrew McCallum. 2007. Learning extractors from unlabeled text using relevant databases. In *IIWeb workshop at AAAI 2007*.

Phil Blunsom and Trevor Cohn. 2006. Discriminative word alignment with conditional random fields. In *ACL*.

Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *EDBT Workshop*, pages 172–183.

Peter Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19:263–311.

Sander Canisius and Caroline Sporleder. 2007. Bootstrapping information extraction from field books. In *EMNLP-CoNLL*.

M. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *ACL*, pages 280–287.

William Cohen, Pradeep Ravikumar, and Stephen Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *IJCAI*.

O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the Web: An experimental study. *Artificial Intelligence*, 165.

D. Freitag and A. McCallum. 1999. Information extraction with HMM and shrinkage. In *AAAI*.

T. Grenager, D. Klein, and C. D. Manning. 2005. Unsupervised learning of field segmentation models for information extraction. In *ACL*.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *HLT-NAACL*.

John Lafferty, Andrew McCallum, and Fernando C N Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, page 282.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Association for Computational Linguistics (ACL)*.

Gideon S. Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of ACL'08*, pages 870–878.

I. R. Mansuri and S. Sarawagi. 2006. Integrating unstructured data into relational databases. In *ICDE*.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *UAI*.

Matthew Michelson and Craig A. Knoblock. 2005. Semantic annotation of unstructured and ungrammatical text. In *IJCAI*, pages 1091–1098.

Matthew Michelson and Craig A. Knoblock. 2008. Creating relational data from unstructured and ungrammatical data sources. *JAIR*, 31:543–590.

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

Fuchun Peng and A. McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In *HLT-NAACL*.

Lawrence R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech processing. *IEEE*, 17:257–286.

Sridhar Ramakrishnan and Sarit Mukherjee. 2004. Taming the unstructured: Creating structured content from partially labeled schematic text sequences. In *CoopIS/DOA/ODBASE*, volume 2, page 909.

Sunita Sarawagi and William W. Cohen. 2004. Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In *KDD*, page 89.

Sunita Sarawagi and William W. Cohen. 2005. Semi-Markov conditional random fields for information extraction. In *NIPS*.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP*, pages 1070–1079.

K. Seymore, A. McCallum, and R. Rosenfeld. 1999. Learning hidden markov model structure for information extraction. In *Proceedings of the AAAI Workshop on ML for IE*.

Benjamin Snyder and Regina Barzilay. 2007. Database-text alignment via structured multi-label classification. In *IJCAI*.

Charles Sutton, Michael Sindelar, and Andrew McCallum. 2006. Reducing weight undertraining in structured discriminative learning. In *HLT-NAACL*.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 2005. A discriminative matching approach to word alignment. In *HLT-EMNLP*, pages 73–80.

Nested Named Entity Recognition

Jenny Rose Finkel and Christopher D. Manning
Computer Science Department
Stanford University
Stanford, CA 94305
{jrfinkel|manning}@cs.stanford.edu

Abstract

Many named entities contain other named entities inside them. Despite this fact, the field of named entity recognition has almost entirely ignored nested named entity recognition, but due to technological, rather than ideological reasons. In this paper, we present a new technique for recognizing nested named entities, by using a discriminative constituency parser. To train the model, we transform each sentence into a tree, with constituents for each named entity (and no other syntactic structure). We present results on both newspaper and biomedical corpora which contain nested named entities. In three out of four sets of experiments, our model outperforms a standard semi-CRF on the more traditional top-level entities. At the same time, we improve the overall F-score by up to 30% over the flat model, which is unable to recover any nested entities.

1 Introduction

Named entity recognition is the task of finding entities, such as people and organizations, in text. Frequently, entities are nested within each other, such as *Bank of China* and *University of Washington*, both *organizations* with nested *locations*. Nested entities are also common in biomedical data, where different biological entities of interest are often composed of one another. In the GENIA corpus (Ohta et al., 2002), which is labeled with entity types such as *protein* and *DNA*, roughly 17% of entities are embedded within another entity. In the AnCora corpus of Spanish and Catalan newspaper text (Martí et al., 2007), nearly half of the entities are embedded. However, work on named entity recognition (NER) has almost entirely ignored nested entities and instead chosen to focus on the outermost entities.

We believe this has largely been for practical, not ideological, reasons. Most corpus designers have chosen to skirt the issue entirely, and have annotated only the topmost entities. The widely used CoNLL (Sang and Meulder, 2003), MUC-6, and MUC-7 NER corpora, composed of American and British newswire, are all flatly annotated. The GENIA corpus contains nested entities, but the JNLPBA 2004 shared task (Collier et al., 2004), which utilized the corpus, removed all embedded entities for the evaluation. To our knowledge, the only shared task which has included nested entities is the SemEval 2007 Task 9 (Márquez et al., 2007b), which used a subset of the AnCora corpus. However, in that task all entities corresponded to particular parts of speech or noun phrases in the provided syntactic structure, and no participant directly addressed the nested nature of the data.

Another reason for the lack of focus on nested NER is technological. The NER task arose in the context of the MUC workshops, as small chunks which could be identified by finite state models or gazetteers. This then led to the widespread use of sequence models, first hidden Markov models, then conditional Markov models (Borthwick, 1999), and, more recently, linear chain conditional random fields (CRFs) (Lafferty et al., 2001). All of these models suffer from an inability to model nested entities.

In this paper we present a novel solution to the problem of nested named entity recognition. Our model explicitly represents the nested structure, allowing entities to be influenced not just by the labels of the words surrounding them, as in a CRF, but also by the entities contained in them, and in which they are contained. We represent each sentence as a parse tree, with the words as leaves, and with phrases corresponding to each entity (and a node which joins the entire sentence). Our trees look just like syntactic constituency trees, such as those in the Penn TreeBank (Marcus et al., 1993),

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 141–150,
Singapore, 6-7 August 2009.

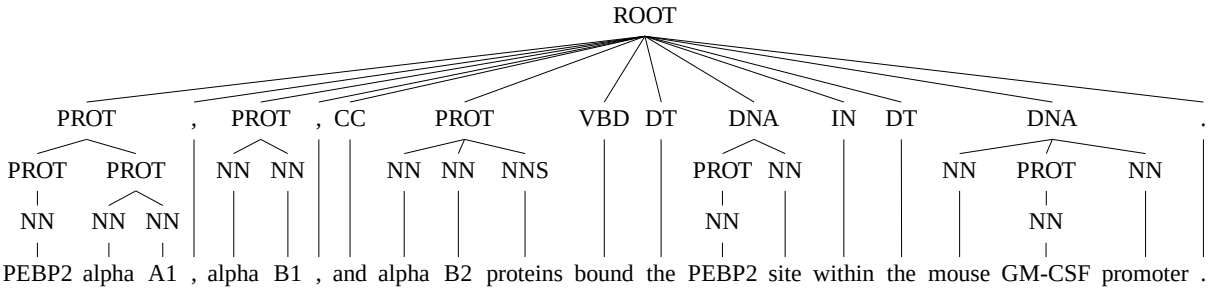


Figure 1: An example of our tree representation over nested named entities. The sentence is from the GENIA corpus. *PROT* is short for *PROTEIN*.

but they tend to be much flatter. This model allows us to include parts of speech in the tree, and therefore to jointly model the named entities and the part of speech tags. Once we have converted our sentences into parse trees, we train a discriminative constituency parser similar to that of (Finkel et al., 2008). We found that on top-level entities, our model does just as well as more conventional methods. When evaluating on *all* entities our model does well, with F-scores ranging from slightly worse than performance on top-level only, to substantially better than top-level only.

2 Related Work

There is a large body of work on named entity recognition, but very little of it addresses nested entities. Early work on the GENIA corpus (Kazama et al., 2002; Tsuruoka and Tsujii, 2003) only worked on the innermost entities. This was soon followed by several attempts at nested NER in GENIA (Shen et al., 2003; Zhang et al., 2004; Zhou et al., 2004) which built hidden Markov models over the innermost named entities, and then used a rule-based post-processing step to identify the named entities containing the innermost entities. Zhou (2006) used a more elaborate model for the innermost entities, but then used the same rule-based post-processing method on the output to identify non-innermost entities. Gu (2006) focused only on proteins and DNA, by building separate binary SVM classifiers for innermost and outermost entities for those two classes.

Several techniques for nested NER in GENIA where presented in (Alex et al., 2007). Their first approach was to layer CRFs, using the output of one as the input to the next. For inside-out layering, the first CRF would identify the innermost entities, the next layer would be over the words and the innermost entities to identify second-level entities, etc. For outside-in layering the first CRF would identify outermost entities, and then successive CRFs would identify increasingly nested entities. They also tried a cascaded approach, with separate CRFs for each entity type. The CRFs would be applied in a specified order, and then each CRF could utilize features derived from the output of previously applied CRFs. This technique has the problem that it cannot identify nested entities of the same type; this happens frequently in the data, such as the nested *proteins* at the beginning of the sentence in Figure 1. They also tried a joint labeling approach, where they trained a single CRF, but the label set was significantly expanded so that a single label would include all of the entities for a particular word. Their best results where from the cascaded approach.

Byrne (2007) took a different approach, on historical archive text. She modified the data by concatenating adjacent tokens (up to length six) into potential entities, and then labeled each concatenated string using the C&C tagger (Curran and Clark, 1999). When labeling a string, the "previous" string was the one-token-shorter string containing all but the last token of the current string. For single tokens the "previous" token was the longest concatenation starting one token earlier.

SemEval 2007 Task 9 (Márquez et al., 2007b) included a nested NER component, as well as noun sense disambiguation and semantic role labeling. However, the parts of speech and syntactic tree were given as part of the input, and named entities were specified as corresponding to noun phrases in the tree, or particular parts of speech. This restriction substantially changes the task. Two groups participated in the shared task, but only one (Márquez et al., 2007a) worked on the named entity component. They used a multi-label AdaBoost.MH algorithm, over phrases in the

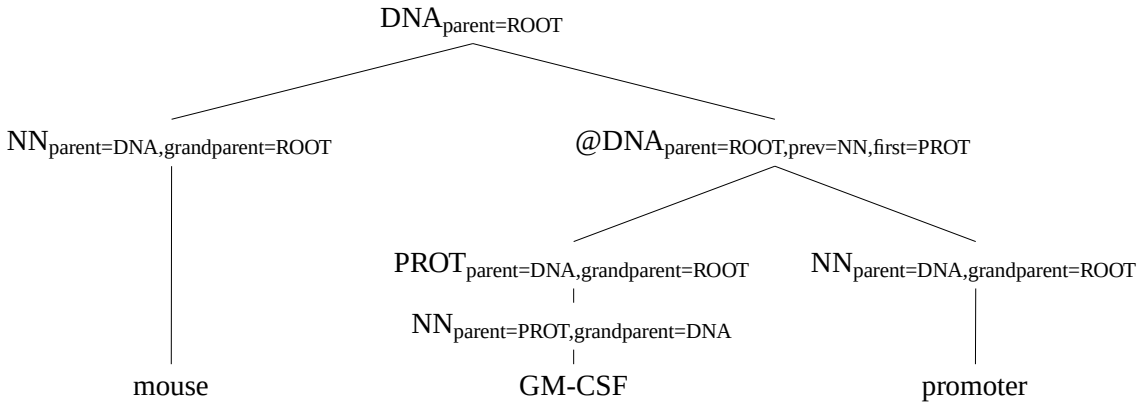


Figure 2: An example of a subtree after it has been annotated and binarized. Features are computed over this representation. An @ indicates a chart parser active state (incomplete constituent).

parse tree which, based on their labels, could potentially be entities.

Finally, McDonald et al. (2005) presented a technique for labeling potentially overlapping segments of text, based on a large margin, multilabel classification algorithm. Their method could be used for nested named entity recognition, but the experiments they performed were on joint (flat) NER and noun phrase chunking.

3 Nested Named Entity Recognition as Parsing

Our model is quite simple – we represent each sentence as a constituency tree, with each named entity corresponding to a phrase in the tree, along with a root node which connects the entire sentence. No additional syntactic structure is represented. We also model the parts of speech as preterminals, and the words themselves as the leaves. See Figure 1 for an example of a named entity tree. Each node is then annotated with both its parent and grandparent labels, which allows the model to learn how entities nest. We binarize our trees in a right-branching manner, and then build features over the labels, unary rules, and binary rules. We also use first-order horizontal Markovization, which allows us to retain some information about the previous node in the binarized rule. See Figure 2 for an example of an annotated and binarized subtree. Once each sentence has been converted into a tree, we train a discriminative constituency parser, based on (Finkel et al., 2008).

It is worth noting that if you use our model on data which does not have any nested entities, then it is precisely equivalent to a semi-CRF (Sarawagi and Cohen, 2004; Andrew, 2006), but with no length restriction on entities. Like a semi-CRF, we are able to define features over entire entities of arbitrary length, instead of just over a small, fixed window of words like a regular linear chain CRF.

We model part of speech tags jointly with the named entities, though the model also works without them. We determine the possible part of speech tags based on distributional similarity clusters. We used Alexander Clarke's software,[1] based on (Clark, 2003), to cluster the words, and then allow each word to be labeled with any part of speech tag seen in the data with any other word in the same cluster. Because the parts of speech are annotated with the parent (and grandparent) labels, they determine what, if any, entity types a word can be labeled with. Many words, such as verbs, cannot be labeled with any entities. We also limit our grammar based on the rules observed in the data. The rules whose children include part of speech tags restrict the possible pairs of adjacent tags. Interestingly, the restrictions imposed by this joint modeling (both observed word/tag pairs and observed rules) actually result in much faster inference (and therefore faster train and test times) than a model over named entities alone. This is different from most work on joint modeling of multiple levels of annotation, which usually results in significantly slower inference.

3.1 Discriminative Constituency Parsing

We train our nested NER model using the same technique as the discriminatively trained, conditional random field-based, CRF-CFG parser of (Finkel et al., 2008). The parser is similar to a

[1] http://www.cs.rhul.ac.uk/home/alexc/RHUL/Downloads.html

Local Features

label_i	$\text{distsim}_i + \text{distsim}_{i-1} + \text{label}_i$
$\text{word}_i + \text{label}_i$	$\text{shape}_i + \text{shape}_{i+1} + \text{label}_i$
$\text{word}_{i-1} + \text{label}_i$	$\text{shape}_{i-1} + \text{shape}_i + \text{label}_i$
$\text{word}_{i+1} + \text{label}_i$	$\text{word}_{i-1} + \text{shape}_i + \text{label}_i$
$\text{distsim}_i + \text{label}_i$	$\text{shape}_i + \text{word}_{i+1} + \text{label}_i$
$\text{distsim}_{i-1} + \text{label}_i$	words in a 5 word window
$\text{distsim}_{i+1} + \text{label}_i$	prefixes up to length 6
$\text{shape}_i + \text{label}_i$	suffixes up to length 6
$\text{shape}_{i-1} + \text{label}_i$	
$\text{shape}_{i+1} + \text{label}_i$	

Pairwise Features

$\text{label}_{i-1} + \text{label}_i$
$\text{word}_i + \text{label}_{i-1} + \text{label}_i$
$\text{word}_{i-1} + \text{label}_{i-1} + \text{label}_i$
$\text{word}_{i+1} + \text{label}_{i-1} + \text{label}_i$
$\text{distsim}_i + \text{label}_{i-1} + \text{label}_i$
$\text{distsim}_{i-1} + \text{label}_{i-1} + \text{label}_i$
$\text{distsim}_{i+1} + \text{label}_{i-1} + \text{label}_i$
$\text{distsim}_{i-1} + \text{distsim}_i + \text{label}_{i-1} + \text{label}_i$
$\text{shape}_i + \text{label}_{i-1} + \text{label}_i$
$\text{shape}_{i-1} + \text{label}_{i-1} + \text{label}_i$
$\text{shape}_{i+1} + \text{label}_{i-1} + \text{label}_i$
$\text{shape}_{i-1} + \text{shape}_i + \text{label}_{i-1} + \text{label}_i$
$\text{shape}_{i-1} + \text{shape}_{i+1} + \text{label}_{i-1} + \text{label}_i$

Table 1: The local and pairwise NER features used in all of our experiments. Consult the text for a full description of all features, which includes feature classes not in this table.

chart-based PCFG parser, except that instead of putting probabilities over rules, it puts *clique potentials* over local subtrees. These unnormalized potentials know what span (and split) the rule is over, and arbitrary features can be defined over the local subtree, the span/split and the words of the sentence. The inside-outside algorithm is run over the clique potentials to produce the partial derivatives and normalizing constant which are necessary for optimizing the log likelihood. Optimization is done by stochastic gradient descent.

The only real drawback to our model is runtime. The algorithm is $O(n^3)$ in sentence length. Training on all of GENIA took approximately 23 hours for the nested model and 16 hours for the semi-CRF. A semi-CRF *with* an entity length restriction, or a regular CRF, would both have been faster. At runtime, the nested model for GENIA tagged about 38 words per second, while the semi-CRF tagged 45 words per second. For comparison, a first-order linear chain CRF trained with similar features on the same data can tag about 4,000 words per second.

4 Features

When designing features, we first made ones similar to the features typically designed for a first-order CRF, and then added features which are not possible in a CRF, but are possible in our enhanced representation. This includes features over entire entities, features which directly model nested entities, and joint features over entities and parts of speech. When features are computed over each label, unary rule, and binary rule, the feature function is aware of the rule span and split.

Each word is labeled with its distributional similarity cluster (*distsim*), and a string indicating orthographic information (*shape*) (Finkel et al., 2005). Subscripts represent word position in the sentence. In addition to those below, we include features for each fully annotated label and rule.

Local named entity features. Local named entity features are over the label for a single word. They are equivalent to the local features in a linear chain CRF. However, unlike in a linear chain CRF, if a word belongs to multiple entities then the local features are computed for each entity. Local features are also computed for words not contained in any entity. Local features are in Table 1.

Pairwise named entity features. Pairwise features are over the labels for adjacent words, and are equivalent to the edge features in a linear chain CRF. They can occur when pairs of words have the same label, or over entity boundaries where the words have different labels. Like with the local features, if a pair of words are contained in, or straddle the border of, multiple entities, then the features are repeated for each. The pairwise features we use are shown in Table 1.

Embedded named entity features. Embedded named entity features occur in binary rules where one entity is the child of another entity. For our embedded features, we replicated the pairwise features, except that the embedded named entity was treated as one of the words, where the "word" (and other annotations) were indicative of the type of entity, and not the actual string that is the entity. For instance, in the subtree in Figure 2, we would compute $word_i + label_{i-1} + label_i$ as *PROT-DNA-DNA* for $i = 18$ (the index of the word *GM-CSF*). The normal pairwise feature at the same po-

GENIA – Testing on All Entities

	# Test Entities	Nested NER Model (train on all entities) Precision	Recall	F_1	Semi-CRF Model (train on top-level entities) Precision	Recall	F_1
Protein	3034	79.04	69.22	73.80	78.63	64.04	70.59
DNA	1222	69.61	61.29	65.19	71.62	57.61	63.85
RNA	103	86.08	66.02	74.73	79.27	63.11	70.27
Cell Line	444	73.82	56.53	64.03	76.59	59.68	67.09
Cell Type	599	68.77	65.44	67.07	72.12	59.60	65.27
Overall	5402	75.39	**65.90**	**70.33**	**76.17**	61.72	68.19

Table 2: Named entity results on GENIA, evaluating on all entities.

GENIA – Testing on Top-level Entities Only

	# Test Entities	Nested NER Model (train on all entities) Precision	Recall	F_1	Semi-CRF Model (train on top-level entities) Precision	Recall	F_1
Protein	2592	78.24	72.42	75.22	76.16	72.61	74.34
DNA	1129	70.40	64.66	67.41	71.21	62.00	66.29
RNA	103	86.08	66.02	74.73	79.27	63.11	70.27
Cell Line	420	75.54	58.81	66.13	76.59	63.10	69.19
Cell Type	537	69.36	70.39	69.87	71.11	65.55	68.22
Overall	4781	**75.22**	**69.02**	**71.99**	74.57	68.27	71.28

Table 3: Named entity results on GENIA, evaluating on only top-level entities.

sition would be *GM-CSF-DNA-DNA*.

Whole entity features. We had four whole entity features: the entire phrase; the preceding and following word; the preceding and following distributional similarity tags; and the preceding distributional similarity tag with the following word.

Local part of speech features. We used the same POS features as (Finkel et al., 2008).

Joint named entity and part of speech features. For the joint features we replicated the POS features, but included the parent of the POS, which either is the innermost entity type, or would indicate that the word is not in any entities.

5 Experiments

We performed two sets of experiments, the first set over biomedical data, and the second over Spanish and Catalan newspaper text. We designed our experiments to show that our model works just as well on outermost entities, the typical NER task, and also works well on nested entities.

5.1 GENIA Experiments

5.1.1 Data

We performed experiments on the GENIA v.3.02 corpus (Ohta et al., 2002). This corpus contains 2000 Medline abstracts (≈500k words), annotated with 36 different kinds of biological entities, and with parts of speech. Previous NER work using this corpus has employed 10-fold cross-validation for evaluation. We wanted to explore different model variations (e.g., level of Markovization, and different sets of distributional similarity clusterings) and feature sets, so we needed to set aside a development set. We split the data by putting the first 90% of sentences into the training set, and the remaining 10% into the test set. This is the exact same split used to evaluate part of speech tagging in (Tsuruoka et al., 2005). For development we used the first half of the data to train, and the next quarter of the data to test.[2] We made the same modifications to the label set as the organizers of the JNLPBA 2004 shared task (Collier et al., 2004). They collapsed all *DNA* subtypes into *DNA*; all *RNA* subtypes into *RNA*; all *protein* subtypes into *protein*; kept *cell line* and *cell type*; and removed all other entities. However, they also removed all embedded entities, while we kept them.

As discussed in Section 3, we annotated each word with a distributional similarity cluster. We used 200 clusters, trained using 200 million words from PubMed abstracts. During development, we found that fewer clusters resulted in slower infer-

[2]This split may seem strange: we had originally intended a 50/25/25 train/dev/test split, until we found the previously used 90/10 split.

JNLPBA 2004 – Testing on Top-level Entities Only

	# Test Entities	Nested NER Model (train on all entities) Precision	Recall	F_1	Semi-CRF Model (train on top-level entities) Precision	Recall	F_1	Zhou & Su (2004) Precision	Recall	F_1
Protein	4944	66.98	74.58	70.57	68.15	62.68	65.30	69.01	79.24	73.77
DNA	1030	62.96	66.50	64.68	65.45	52.23	58.10	66.84	73.11	69.83
RNA	115	63.06	60.87	61.95	64.55	61.74	63.11	64.66	63.56	64.10
Cell line	487	49.92	60.78	54.81	49.61	52.16	50.85	53.85	65.80	59.23
Cell type	1858	75.12	65.34	69.89	73.29	55.81	63.37	78.06	72.41	75.13
Overall	8434	66.78	70.57	68.62	67.50	59.27	63.12	**69.42**	**75.99**	**72.55**

Table 4: Named entity results on the JNLPBA 2004 shared task data. Zhou and Su (2004) was the best system at the shared task, and is still state-of-the-art on the dataset.

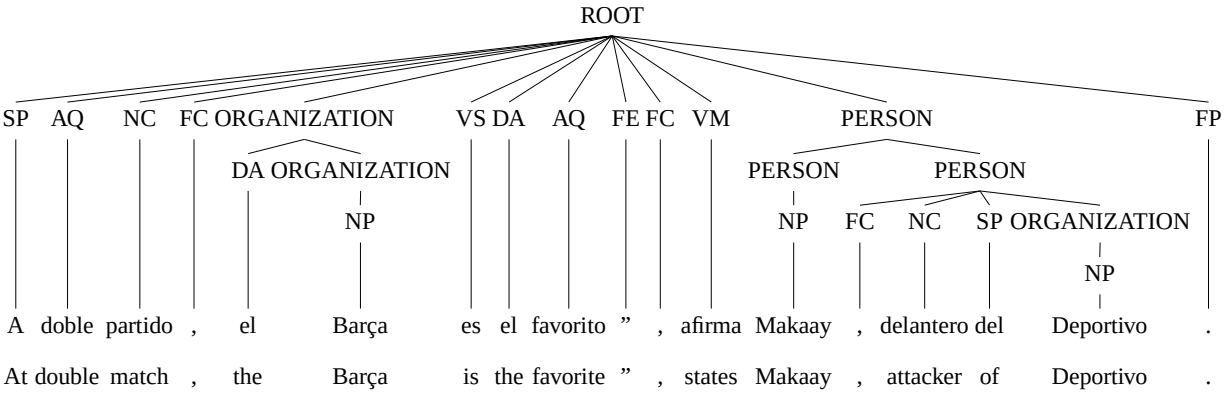


Figure 3: An example sentence from the AnCora corpus, along with its English translation.

ence with no improvement in performance.

5.1.2 Experimental Setup

We ran several sets of experiments, varying between all entities, or just top-level entities, for training and testing. As discussed in Section 3, if we train on just top-level entities then the model is equivalent to a semi-CRF. Semi-CRFs are state-of-the-art and provide a good baseline for performance on just the top-level entities. Semi-CRFs are strictly better than regular, linear chain CRFs, because they can use all of the features and strucutre of a linear chain CRF, but also utilize whole-entity features (Andrew, 2006). We also evaluated the semi-CRF model on all entities. This may seem like an unfair evaluation, because the semi-CRF has no way of recovering the nested entities, but we wanted to illustrate just how much information is lost when using a flat representation.

5.1.3 Results

Our named entity results when evaluating on all entities are shown in Table 2 and when evaluating on only top-level entities are shown in Table 3. Our nested model outperforms the flat semi-CRF on both top-level entities and all entities.

While not our main focus, we also evaluated our models on parts of speech. The model trained on just top level entities achieved POS accuracy of 97.37%, and the one trained on all entities achieved 97.25% accuracy. The GENIA tagger (Tsuruoka et al., 2005) achieves 98.49% accuracy using the same train/test split.

5.1.4 Additional JNLPBA 2004 Experiments

Because we could not compare our results on the NER portion of the GENIA corpus with any other work, we also evaluated on the JNLPBA corpus. This corpus was used in a shared task for the BioNLP workshop at Coling in 2004 (Collier et al., 2004). They used the entire GENIA corpus for training, and modified the label set as discussed in Section 5.1.1. They also removed all embedded entities, and kept only the top-level ones. They then annotated new data for the test set. This dataset has no nested entities, but because the training data is GENIA we can still train our model on the data annotated with nested entities, and then evaluate on their test data by ignoring all embedded entities found by our named entity recognizer.

AnCora Spanish – Testing on All Entities

	# Test Entities	Nested NER Model (train on all entities) Precision	Recall	F_1	Semi-CRF Model (train on top-level entities) Precision	Recall	F_1
Person	1778	65.29	78.91	71.45	75.10	32.73	45.59
Organization	2137	86.43	56.90	68.62	47.02	26.20	33.65
Location	1050	78.66	46.00	58.05	84.94	13.43	23.19
Date	568	87.13	83.45	85.25	79.43	29.23	42.73
Number	991	81.51	80.52	81.02	66.27	28.15	39.52
Other	512	17.90	64.65	28.04	10.77	16.60	13.07
Overall	7036	**62.38**	**66.87**	**64.55**	51.06	25.77	34.25

Table 5: Named entity results on the Spanish portion of AnCora, evaluating on all entities.

AnCora Spanish – Testing on Top-level Entities Only

	# Test Entities	Nested NER Model (train on all entities) Precision	Recall	F_1	Semi-CRF Model (train on top-level entities) Precision	Recall	F_1
Person	1050	57.42	66.67	61.70	71.23	52.57	60.49
Organization	1060	77.38	40.66	53.31	44.33	49.81	46.91
Location	279	72.49	36.04	48.15	79.52	24.40	37.34
Date	290	72.29	57.59	64.11	71.77	51.72	60.12
Number	519	57.17	49.90	53.29	54.87	44.51	49.15
Other	541	11.30	38.35	17.46	9.51	26.88	14.04
Overall	3739	**50.57**	**49.72**	**50.14**	46.07	44.61	45.76

Table 6: Named entity results on the Spanish portion of AnCora, evaluating on only top-level entities.

This experiment allows us to show that our named entity recognizer works well on top-level entities, by comparing it with prior work. Our model also produces part of speech tags, but the test data is not annotated with POS tags, so we cannot show POS tagging results on this dataset.

One difficulty we had with the JNLPBA experiments was with tokenization. The version of GENIA distributed for the shared task is tokenized differently from the original GENIA corpus, but we needed to train on the original corpus as it is the only version with nested entities. We tried our best to retokenize the original corpus to match the distributed data, but did not have complete success. It is worth noting that the data is actually tokenized in a manner which allows a small amount of "cheating." Normally, hyphenated words, such as *LPS-induced*, are tokenized as one word. However, if the portion of the word before the hyphen is in an entity, and the part after is not, such as *BCR-induced*, then the word is split into two tokens: *BCR* and *-induced*. Therefore, when a word starts with a hyphen it is a strong indicator that the prior word and it span the right boundary of an entity. Because the train and test data for the shared task do not contain nested entities, fewer words are split in this manner than in the original data. We did not intentionally exploit this fact in our feature design, but it is probable that some of our orthographic features "learned" this fact anyway. This probably harmed our results overall, because some hyphenated words, which straddled boundaries in nested entities and would have been split in the original corpus (and were split in our training data), were not split in the test data, prohibiting our model from properly identifying them.

For this experiment, we retrained our model on the entire, retokenized, GENIA corpus. We also retrained the distributional similarity model on the retokenized data. Once again, we trained one model on the nested data, and one on just the top-level entities, so that we can compare performance of both models on the top-level entities. Our full results are shown in Table 4, along with the current state-of-the-art (Zhou and Su, 2004). Besides the tokenization issues harming our performance, Zhou and Su (2004) also employed clever post-processing to improve their results.

5.2 AnCora Experiments

5.2.1 Data

We performed experiments on the NER portion of AnCora (Martí et al., 2007). This corpus has Spanish and Catalan portions, and we evaluated on both. The data is also annotated with parts of speech, parse trees, semantic roles and word

AnCora Catalan – Testing on All Entities

	# Test Entities	Nested NER Model (train all entities) Precision	Recall	F_1	Semi-CRF Model (train top-level entities only) Precision	Recall	F_1
Person	1303	89.01	50.35	64.31	70.08	46.20	55.69
Organization	1781	68.95	83.77	75.64	65.32	41.77	50.96
Location	1282	76.78	72.46	74.56	75.49	36.04	48.79
Date	606	84.27	81.35	82.79	70.87	38.94	50.27
Number	1128	86.55	83.87	85.19	75.74	38.74	51.26
Other	596	85.48	8.89	16.11	64.91	6.21	11.33
Overall	6696	**78.09**	**68.23**	**72.83**	70.39	37.60	49.02

Table 7: Named entity results on the Catalan portion of AnCora, evaluating on all entities.

AnCora Catalan – Testing on Top-level Entities Only

	# Test Entities	Nested NER Model (train all entities) Precision	Recall	F_1	Semi-CRF Model (train top-level entities only) Precision	Recall	F_1
Person	801	67.44	47.32	55.61	62.63	67.17	64.82
Organization	899	52.21	74.86	61.52	57.68	73.08	64.47
Location	659	54.86	67.68	60.60	62.42	57.97	60.11
Date	296	62.54	66.55	64.48	59.46	66.89	62.96
Number	528	62.35	70.27	66.07	63.08	68.94	65.88
Other	342	49.12	8.19	14.04	45.61	7.60	13.03
Overall	3525	57.67	59.40	58.52	**60.53**	**61.42**	**60.97**

Table 8: Named entity results on the Catalan portion of AnCora, evaluating on only top-level entities.

senses. The corpus annotators made a distinction between *strong* and *weak* entities. They define *strong* named entities as "a word, a number, a date, or a string of words that refer to a single individual entity in the real world." If a strong NE contains multiple words, it is collapsed into a single token. *Weak* named entities, "consist of a noun phrase, being it simple or complex" and must contain a *strong* entity. Figure 3 shows an example from the corpus with both strong and weak entities. The entity types present are *person*, *location*, *organization*, *date*, *number*, and *other*. Weak entities are very prevalent; 47.1% of entities are embedded.

For Spanish, files starting with 7–9 were the test set, 5–6 were the development test set, and the remainder were the development train set. For Catalan, files starting with 8–9 were the test set, 6–7 were the development test set, and the remainder were the development train set. For both, the development train and test sets were combined to form the final train set. We removed sentences longer than 80 words. Spanish has 15,591 training sentences, and Catalan has 14,906.

5.2.2 Experimental Setup

The parts of speech provided in the data include detailed morphological information, using a similar annotation scheme to the Prague TreeBank (Hana and Hanová, 2002). There are around 250 possible tags, and experiments on the development data with the full tagset where unsuccessful. We removed all but the first two characters of each POS tag, resulting in a set of 57 tags which more closely resembles that of the Penn TreeBank (Marcus et al., 1993). All reported results use our modified version of the POS tag set.

We took only the words as input, none of the extra annotations. For both languages we trained a 200 cluster distributional similarity model over the words in the corpus. We performed the same set of experiments on AnCora as we did on GENIA.

5.2.3 Results and Discussion

The full results for Spanish when testing on all entities are shown in Table 5, and for only top-level entities are shown in Table 6. For part of speech tagging, the nested model achieved 95.93% accuracy, compared with 95.60% for the flatly trained model. The full results for Catalan when testing on all entities are shown in Table 7, and for only top-level entities are shown in Table 8. POS tagging results were even closer on Catalan: 96.62% for the nested model, and 96.59% for the flat model.

It is not surprising that the models trained on all entities do significantly better than the flatly trained models when testing on all entities. The

story is a little less clear when testing on just top-level entities. In this case, the nested model does 4.38% better than the flat model on the Spanish data, but 2.45% worse on the Catalan data. The overall picture is the same as for GENIA: modeling the nested entities does not, on average, reduce performance on the top-level entities, but a nested entity model does substantially better when evaluated on all entities.

6 Conclusions

We presented a discriminative parsing-based method for nested named entity recognition, which does well on both top-level and nested entities. The only real drawback to our method is that it is slower than common flat techniques. While most NER corpus designers have defenestrated embedded entities, we hope that in the future this will not continue, as large amounts of information are lost due to this design decision.

Acknowledgements

Thanks to Mihai Surdeanu for help with the AnCora data. The first author was supported by a Stanford Graduate Fellowship. This paper is based on work funded in part by the Defense Advanced Research Projects Agency through IBM. The content does not necessarily reflect the views of the U.S. Government, and no official endorsement should be inferred.

References

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *BioNLP Workshop at ACL 2007*, pages 65–72.

Galen Andrew. 2006. A hybrid markov/semi-markov conditional random field for sequence segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2006).*

A. Borthwick. 1999. *A Maximum Entropy Approach to Named Entity Recognition*. Ph.D. thesis, New York University.

Kate Byrne. 2007. Nested named entity recognition in historical archive text. In *ICSC '07: Proceedings of the International Conference on Semantic Computing*, pages 589–596.

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

Nigel Collier, J. Kim, Y. Tateisi, T. Ohta, and Y. Tsuruoka, editors. 2004. *Proceedings of the International Joint Workshop on NLP in Biomedicine and its Applications*.

J. R. Curran and S. Clark. 1999. Language independent NER using a maximum entropy tagger. In *CoNLL 1999*, pages 164–167.

Jenny Finkel, Shipra Dingare, Christopher Manning, Malvina Nissim, Beatrice Alex, and Claire Grover. 2005. Exploring the boundaries: Gene and protein identification in biomedical text. In *BMC Bioinformatics 6 (Suppl. 1)*.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based conditional random field parsing. In *ACL/HLT-2008*.

Baohua Gu. 2006. Recognizing nested named entities in GENIA corpus. In *BioNLP Workshop at HLT-NAACL 2006*, pages 112–113.

Jiří Hana and Hana Hanová. 2002. Manual for morphological annotation. Technical Report TR-2002-14, UK MFF CKL.

Jun'ichi Kazama, Takaki Makino, Yoshihiro Ohta, and Jun'ichi Tsujii. 2002. Tuning support vector machines for biomedical named entity recognition. In *Proceedings of the Workshop on Natural Language Processing in the Biomedical Domain (ACL 2002)*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

L. Márquez, L. Padrè, M. Surdeanu, and L. Villarejo. 2007a. UPC: Experiments with joint learning within semeval task 9. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*.

L. Márquez, L. Villarejo, M.A. Martí, and M. Taulè. 2007b. Semeval-2007 task 09: Multilevel semantic annotation of Catalan and Spanish. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*.

M.A. Martí, M. Taulè, M. Bertran, and L. Márquez. 2007. Ancora: Multilingual and multilevel annotated corpora. MS, Universitat de Barcelona.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Flexible text segmentation with structured multilabel classification. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 987–994.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The GENIA corpus: an annotated research abstract corpus in molecular biology domain. In *Proceedings of the second international conference on Human Language Technology Research*, pages 82–86.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*.

Sunita Sarawagi and William W. Cohen. 2004. Semi-markov conditional random fields for information extraction. In *In Advances in Neural Information Processing Systems 17*, pages 1185–1192.

Dan Shen, Jie Zhang, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2003. Effective adaptation of a hidden markov model-based named entity recognizer for biomedical domain. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine*. Association for Computational Linguistics (ACL 2003).

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2003. Boosting precision and recall of dictionary-based protein name recognition. In *Proceedings of the ACL-03 Workshop on Natural Language Processing in Biomedicine*, pages 41–48.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. 2005. Developing a robust part-of-speech tagger for biomedical text. In *Advances in Informatics - 10th Panhellenic Conference on Informatics, LNCS 3746*, pages 382–392.

Jie Zhang, Dan Shen, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2004. Enhancing HMM-based biomedical named entity recognition by studying special phenomena. *Journal of Biomedical Informatics*, 37(6):411–422.

GuoDong Zhou and Jian Su. 2004. Exploring deep knowledge resources in biomedical name recognition. In *Joint Workshop on Natural Language Processing in Biomedicine and Its Applications at Coling 2004*.

Guodong Zhou, Jie Zhang, Jian Su, Dan Shen, and Chewlim Tan. 2004. Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics*, 20(7):1178–1190.

Guodong Zhou. 2006. Recognizing names in biomedical texts using mutual information independence model and SVM plus sigmoid. *International Journal of Medical Informatics*, 75:456–467.

A Unified Model of Phrasal and Sentential Evidence for Information Extraction

Siddharth Patwardhan and **Ellen Riloff**
School of Computing
University of Utah
Salt Lake City, UT 84112
{sidd,riloff}@cs.utah.edu

Abstract

Information Extraction (IE) systems that extract role fillers for events typically look at the local context surrounding a phrase when deciding whether to extract it. Often, however, role fillers occur in clauses that are not directly linked to an event word. We present a new model for event extraction that jointly considers both the local context around a phrase along with the wider sentential context in a probabilistic framework. Our approach uses a *sentential event recognizer* and a *plausible role-filler recognizer* that is conditioned on event sentences. We evaluate our system on two IE data sets and show that our model performs well in comparison to existing IE systems that rely on local phrasal context.

1 Introduction

Information Extraction (IE) systems typically use extraction patterns (e.g., Soderland et al. (1995), Riloff (1996), Yangarber et al. (2000), Califf and Mooney (2003)) or classifiers (e.g., Freitag (1998), Freitag and McCallum (2000), Chieu et al. (2003), Bunescu and Mooney (2004)) to extract role fillers for events. Most IE systems consider only the immediate context surrounding a phrase when deciding whether to extract it. For tasks such as named entity recognition, immediate context is usually sufficient. But for more complex tasks, such as event extraction, a larger field of view is often needed to understand how facts tie together.

Most IE systems are designed to identify role fillers that appear as arguments to event verbs or nouns, either explicitly via syntactic relations or implicitly via proximity (e.g., *John murdered Tom* or *the murder of Tom by John*). But many facts are presented in clauses that do not contain event words, requiring discourse relations or deep structural analysis to associate the facts with event roles. For example, consider the sentences below:

Seven people *have died*

. . . and 30 were injured in India after terrorists launched an attack on the Taj Hotel.

. . . in Mexico City and its surrounding suburbs in a Swine Flu outbreak.

. . . after a tractor-trailer collided with a bus in Arkansas.

Two bridges *were destroyed*

. . . in Baghdad last night in a resurgence of bomb attacks in the capital city.

. . . and $50 million in damage was caused by a hurricane that hit Miami on Friday.

. . . to make way for modern, safer bridges that will be constructed early next year.

These examples illustrate a common phenomenon in text where information is not explicitly stated as filling an event role, but readers have no trouble making this inference. The role fillers above (***seven people***, ***two bridges***) occur as arguments to verbs that reveal state information (death, destruction) but are not event-specific (i.e., death and destruction can result from a wide variety of incident types). IE systems often fail to extract these role fillers because these systems do not recognize the immediate context as being relevant to the specific type of event that they are looking for.

We propose a new model for information extraction that incorporates both phrasal and sentential evidence in a unified framework. Our unified probabilistic model, called GLACIER, consists of two components: a model for *sentential event recognition* and a model for recognizing *plausible role fillers*. The Sentential Event Recognizer offers a probabilistic assessment of whether a sentence is discussing a domain-relevant event. The

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 151–160,
Singapore, 6-7 August 2009.

Plausible Role-Filler Recognizer is then conditioned to identify phrases as role fillers based upon the assumption that the surrounding context is discussing a relevant event. This unified probabilistic model allows the two components to jointly make decisions based upon both the local evidence surrounding each phrase and the "peripheral vision" afforded by the sentential event recognizer.

This paper is organized as follows. Section 2 positions our research with respect to related work. Section 3 presents our unified probabilistic model for information extraction. Section 4 shows experimental results on two IE data sets, and Section 5 discusses directions for future work.

2 Related Work

Many event extraction systems rely heavily on the local context around words or phrases that are candidates for extraction. Some systems use extraction patterns (Soderland et al., 1995; Riloff, 1996; Yangarber et al., 2000; Califf and Mooney, 2003), which represent the immediate contexts surrounding candidate extractions. Similarly, classifier-based approaches (Freitag, 1998; Freitag and McCallum, 2000; Chieu et al., 2003; Bunescu and Mooney, 2004) rely on features in the immediate context of the candidate extractions. Our work seeks to incorporate additional context into IE.

Indeed, several recent approaches have shown the need for global information to improve IE performance. Maslennikov and Chua (2007) use discourse trees and local syntactic dependencies in a pattern-based framework to incorporate wider context. Finkel et al. (2005) and Ji and Grishman (2008) incorporate global information by enforcing event role or label consistency over a document or across related documents. In contrast, our approach simply creates a richer IE model for individual extractions by expanding the "field of view" to include the surrounding sentence.

The two components of the unified model presented in this paper are somewhat similar to our previous work (Patwardhan and Riloff, 2007), where we employ a relevant region identification phase prior to pattern-based extraction. In that work we adopted a pipeline paradigm, where a classifier identifies relevant sentences and only those sentences are fed to the extraction module. Our unified probabilistic model described in this paper does not draw a hard line between relevant and irrelevant sentences, but gently balances the influence of both local and sentential contexts through probability estimates.

3 A Unified IE Model that Combines Phrasal and Sentential Evidence

We introduce a probabilistic model for event-based IE that balances the influence of two kinds of contextual information. Our goal is to create a model that has the flexibility to make extraction decisions based upon strong evidence from the local context, or strong evidence from the wider context coupled with a more general local context. For example, some phrases explicitly refer to an event, so they almost certainly warrant extraction regardless of the wider context (e.g., *terrorists launched an attack*).[1] In contrast, some phrases are potentially relevant but too general to warrant extraction on their own (e.g., *people died* could be the result of different incident types). If we are confident that the sentence discusses an event of interest, however, then such phrases could be reliably extracted.

Our unified model for IE (GLACIER) combines two types of contextual information by incorporating it into a probabilistic framework. To determine whether a noun phrase instance NP_i should be extracted as a filler for an event role, GLACIER computes the joint probability that NP_i:

(1) appears in an event sentence, and
(2) is a legitimate filler for the event role.

Thus, GLACIER is designed for noun phrase extraction and, mathematically, its decisions are based on the following joint probability:

$$P(EvSent(S_{NP_i}), PlausFillr(NP_i))$$

where S_{NP_i} is the sentence containing noun phrase NP_i. This probability estimate is based on contextual features F appearing within S_{NP_i} and in the local context of NP_i. Including F in the joint probability, and applying the product rule, we can split our probability into two components:

$$\begin{aligned} P(EvSent(S_{NP_i}), PlausFillr(NP_i)|F) = \\ P(EvSent(S_{NP_i})|F) \\ * \quad P(PlausFillr(NP_i)|EvSent(S_{NP_i}), F) \end{aligned}$$

These two probability components, in the expression above, form the basis of the two modules in

[1] There are always exceptions of course, such as hypothetical statements, but they are relatively uncommon.

our IE system – the *sentential event recognizer* and the *plausible role-filler recognizer*. In arriving at a decision to extract a noun phrase, our unified model for IE uses these modules to estimate the two probabilities based on the set of contextual features F. Note that having these two probability components allows the system to gently balance the influence from the sentential and phrasal contexts, without having to make hard decisions about sentence relevance or phrases in isolation.

In this system, the sentential event recognizer is embodied in the probability component $P(EvSent(S_{NP_i})|F)$. This is essentially the probability of a sentence describing a relevant event. Similarly, the plausible role-filler recognizer is embodied by the probability $P(PlausFillr(NP_i)|EvSent(S_{NP_i}), F)$. This component, therefore, estimates the probability that a noun phrase fills a specific event role, *assuming that the noun phrase occurs in an event sentence*. Many different techniques could be used to produce these probability estimates. In the rest of this section, we present the specific models that we used for each of these components.

3.1 Plausible Role-Filler Recognizer

The plausible role-filler recognizer is similar to most traditional IE systems, where the goal is to determine whether a noun phrase can be a legitimate filler for a specific type of event role based on its local context. Pattern-based approaches match the context surrounding a phrase using lexico-syntactic patterns or rules. However, most of these approaches do not produce probability estimates for the extractions. Classifier-based approaches use machine learning classifiers to make extraction decisions, based on features associated with the local context. Any classifier that can generate probability estimates, or similar confidence values, could be plugged into our model.

In our work, we use a Naïve Bayes classifier as our plausible role-filler recognizer. The probabilities are computed using a generative Naïve Bayes framework, based on local contextual features surrounding a noun phrase. These clues include lexical matches, semantic features, and syntactic relations, and will be described in more detail in Section 3.3. The Naïve Bayes (NB) plausible role-filler recognizer is defined as follows:

$$P(PlausFillr(NP_i)|EvSent(S_{NP_i}), F) = \frac{1}{Z} P(PlausFillr(NP_i)|EvSent(S_{NP_i})) * \prod_{f_i \in F} P(f_i|PlausFillr(NP_i), EvSent(S_{NP_i}))$$

where F is the set of local contextual features and Z is the normalizing constant. The prior $P(PlausFillr(NP_i)|EvSent(S_{NP_i}))$ is estimated from the fraction of role fillers in the training data. The product term in the equation is the likelihood, which makes the simplifying assumption that all of the features in F are independent of one another. It is important to note that these probabilities are conditioned on the noun phrase NP_i appearing in an event sentence.

Most IE systems need to extract several different types of role fillers for each event. For instance, to extract information about terrorist incidents a system may extract the names of perpetrators, victims, targets, and weapons. We create a separate IE model for each type of event role. To construct a unified IE model for an event role, we must specifically create a plausible role-filler recognizer for that event role, but we can use a single sentential event recognizer for all of the role filler types.

3.2 Sentential Event Recognizer

The task at hand for the sentential event recognizer is to analyze features in a sentence and estimate the probability that the sentence is discussing a relevant event. This is very similar to the task performed by text classification systems, with some minor differences. Firstly, we are dealing with the classification of sentences, as opposed to entire documents. Secondly, we need to generate a probability estimate of the "class", and not just a class label. Like the plausible role-filler recognizer, here too we employ machine learning classifiers to estimate the desired probabilities.

3.2.1 Naïve Bayes Event Recognizer

Since Naïve Bayes classifiers estimate class probabilities, we employ such a classifier to create a sentential event recognizer:

$$P(EvSent(S_{NP_i})|F) = \frac{1}{Z} P(EvSent(S_{NP_i})) * \prod_{f_i \in F} P(f_i|EvSent(S_{NP_i}))$$

where Z is the normalizing constant and F is the set of contextual features in the sentence. The

prior $P(EvSent_{S(NP_i)})$ is obtained from the ratio of event and non-event sentences in the training data. The product term in the equation is the likelihood, which makes the simplifying assumption that the features in F are independent of one another. The features used by the model will be described in Section 3.3.

A known issue with Naïve Bayes classifiers is that, even though their classification accuracy is often quite reasonable, their probability estimates are often poor (Domingos and Pazzani, 1996; Zadrozny and Elkan, 2001; Manning et al., 2008). The problem is that these classifiers tend to overestimate the probability of the predicted class, resulting in a situation where most probability estimates from the classifier tend to be either extremely close to 0.0 or extremely close to 1.0. We observed this problem in our classifier too, so we decided to explore an additional model to estimate probabilities for the sentential event recognizer. This second model, based on SVMs, is described next.

3.2.2 SVM Event Recognizer

Given the all-or-nothing nature of the probability estimates that we observed from the Naïve Bayes model, we decided to try using a Support Vector Machine (SVM) (Vapnik, 1995; Joachims, 1998) classifier as an alternative to Naïve Bayes. One of the issues with doing this is that SVMs are not probabilistic classifiers. SVMs make classification decisions using on a *decision boundary* defined by *support vectors* identified during training. A decision function is applied to unseen test examples to determine which side of the decision boundary those examples lie. While the values obtained from the decision function only indicate class assignments for the examples, we used these values to produce confidence scores for our sentential event recognizer.

To produce a confidence score from the SVM classifier, we take the values generated by the decision function for each test instance and normalize them based on the minimum and maximum values produced across all of the test instances. This normalization process produces values between 0 and 1 that we use as a rough indicator of the confidence in the SVM's classification. We observed that we could effect a consistent recall/precision trade-off by using these values as thresholds for classification decisions, which suggests that this approach worked reasonably well for our task.

3.3 Contextual Features

We used a variety of contextual features in both components of our system. The plausible role-filler recognizer uses the following types of features for each candidate noun phrase NP_i: *lexical head* of NP_i, *semantic class* of NP_i's lexical head, *named entity tags* associated with NP_i and *lexico-syntactic patterns* that represent the local context surrounding NP_i. The feature set is automatically generated from the texts. Each feature is assigned a binary value for each instance, indicating either the presence or absence of the feature.

The *named-entity* features are generated by the freely available Stanford NER tagger (Finkel et al., 2005). We use the pre-trained NER model that comes with the software to identify person, organization and location names. The syntactic and semantic features are generated by the Sundance/AutoSlog system (Riloff and Phillips, 2004). We use the Sundance shallow parser to identify lexical heads, and use its semantic dictionaries to assign semantic features to words. The AutoSlog pattern generator (Riloff, 1996) is used to create the *lexico-syntactic pattern* features that capture local context around each noun phrase.

Our training sets produce a very large number of features, which initially bogged down our classifiers. Consequently, we reduced the size of the feature set by discarding all features that appeared four times or less in the training set.

Our sentential event recognizer uses the same contextual features as the plausible role-filler recognizer, except that features are generated for every NP in the sentence. In addition, it uses three types of sentence-level features: *sentence length*, *bag of words*, and *verb tense*, which are also binary features. We have two binary *sentence length* features indicating that the sentence is long (greater than 35 words) or is short (shorter than 5 words). Additionally, all of the words in each sentence in the training data are generated as *bag of words* features for the sentential model. Finally, we generate *verb tense* features from all verbs appearing in each sentence. Here too we apply a frequency cutoff and eliminate all features that appear four times or less in the training data.

4 IE Evaluation

4.1 Data Sets

We evaluated the performance of our IE system on two data sets: the MUC-4 terrorism corpus (Sund-

heim, 1992), and a ProMed disease outbreaks corpus (Phillips and Riloff, 2007; Patwardhan and Riloff, 2007). The MUC-4 data set is a standard IE benchmark collection of news stories about terrorist events. It contains 1700 documents divided into 1300 development (DEV) texts, and four test sets of 100 texts each (TST1, TST2, TST3, and TST4). Unless otherwise stated, our experiments adopted the same training/test split used in previous research: the 1300 DEV texts for training, 200 texts (TST1+TST2) for tuning, and 200 texts (TST3+TST4) as the blind test set. We evaluated our system on five MUC-4 string roles: *perpetrator individuals*, *perpetrator organizations*, *physical targets*, *victims*, and *weapons*.

The ProMed corpus consists of 120 documents obtained from ProMed-mail[2], a freely accessible global electronic reporting system for outbreaks of diseases. These 120 documents are paired with corresponding answer key templates. Unless otherwise noted, all of our experiments on this data set used 5-fold cross validation. We extracted two types of event roles: *diseases* and *victims*[3].

Unlike some other IE data sets, many of the texts in these collections do not describe a relevant event. Only about half of the MUC-4 articles describe a specific terrorist incident[4], and only about 80% of the ProMed articles describe a disease outbreak. The answer keys for the irrelevant documents are therefore empty. IE systems are especially susceptible to false hits when they can be given texts that contain no relevant events.

The complete IE task involves the creation of answer key templates, one template per incident (many documents in our data sets describe multiple events). Our work focuses on accurately extracting the facts from the text and not on template generation per se (e.g., we are not concerned with coreference resolution or which extraction belongs in which template). Consequently, our experiments evaluate the accuracy of the extractions individually. We used *head noun* scoring, where an extraction is considered to be correct if its head noun matches the head noun in the answer key.[5]

[2]http://www.promedmail.org

[3]The "victims" can be people, animals, or plants.

[4]With respect to the definition of terrorist incidents in the MUC-4 guidelines (Sundheim, 1992).

[5]Pronouns were discarded from both the system responses and the answer keys since we do not perform coreference resolution. Duplicate extractions (e.g., the same string extracted multiple times from the same document) were conflated before being scored, so they count as just one hit or one miss.

4.2 Baselines

We generated three baselines to use as comparisons with our IE system. As our first baseline, we used AutoSlog-TS (Riloff, 1996), which is a weakly-supervised, pattern-based IE system available as part of the Sundance/AutoSlog software package (Riloff and Phillips, 2004). Our previous work in event-based IE (Patwardhan and Riloff, 2007) also used a pattern-based approach that applied *semantic affinity* patterns to relevant regions in text. We use this system as our second baseline. As a third baseline, we trained a Naïve Bayes IE classifier that is analogous to the plausible role-filler recognizer in our unified IE model, except that this baseline system is not conditioned on the assumption of having an event sentence. Consequently, this baseline NB classifier is akin to a traditional supervised learning-based IE system that uses only local contextual features to make extraction decisions. Formally, the baseline NB classifier uses the formula:

$$P(PlausFillr(NP_i)|F) = \frac{1}{Z}P(PlausFillr(NP_i)) * \prod_{f_i \in F} P(f_i|PlausFillr(NP_i))$$

where F is the set of local features, $P(PlausFillr(NP_i))$ is the prior probability, and Z is the normalizing constant. We used the Weka (Witten and Frank, 2005) implementation of Naïve Bayes for this baseline NB system.

New Jersey, February, 26. An outbreak of Ebola has been confirmed in Mercer County, New Jersey. Five teenage boys appear to have contracted the deadly virus from an unknown source. The CDC is investigating the cases and is taking measures to prevent the spread...

Disease:	Ebola
Victims:	Five teenage boys
Location:	Mercer County, New Jersey
Date:	February 26

Figure 1: A *Disease Outbreak* Event Template

Both the MUC-4 and ProMed data sets have separate answer keys rather than annotated source documents. Figure 1 shows an example of a document and its corresponding answer key template. To train the baseline NB system, we identify all instances of each answer key string in the source document and consider every instance a positive training example. This produces noisy training data, however, because some instances occur in

	PerpInd			PerpOrg			Target			Victim			Weapon		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
AutoSlog-TS	.33	.49	.40	.52	.33	.41	.54	.59	.56	.49	.54	.51	.38	.44	.41
Sem Affinity	.48	.39	.43	.36	.58	.45	.56	.46	.50	.46	.44	.45	.53	.46	.50
NB .50	.36	.34	.35	.35	.46	.40	.53	.49	.51	.50	.50	.50	1.00	.05	.10
NB .70	.41	.25	.31	.43	.31	.36	.58	.42	.48	.58	.37	.45	1.00	.04	.07
NB .90	.51	.17	.25	.56	.15	.24	.67	.30	.41	.75	.23	.36	1.00	.02	.04

Table 1: Baseline Results on MUC-4

	Disease			Victim		
	P	R	F	P	R	F
AutoSlog-TS	.33	.60	.43	.36	.49	.41
Sem Affinity	.31	.49	.38	.41	.47	.44
NB .50	.20	.73	.31	.29	.56	.39
NB .70	.23	.67	.34	.37	.52	.44
NB .90	.34	.59	.43	.47	.39	.43

Table 2: Baseline Results on ProMed

undesirable contexts. For example, if the string "man" appears in an answer key as a victim, one instance of "man" may refer to the actual victim in an event sentence, while another instance of "man" may occur in a non-event context (e.g., background information) or may refer to a completely different person.

We report three evaluation metrics in our experiments: precision (P), recall (R), and F-score (F), where recall and precision are equally weighted. For the Naïve Bayes classifier, the natural threshold for distinguishing between positive and negative classes is 0.5, but we also evaluated this classifier with thresholds of 0.7 and 0.9 to see if we could effect a recall/precision trade-off. Tables 1 and 2 present the results of our three baseline systems. The NB classifier performs comparably to AutoSlog-TS and Semantic Affinity on most event roles, although a threshold of 0.90 is needed to reach comparable performance on ProMed. The relatively low numbers across the board indicate that these corpora are challenging, but these results suggest that our plausible role-filler recognizer is competitive with other existing IE systems. In Section 4.4 we will show how our unified IE model compares to these baselines. But before that (in the next section) we evaluate the quality of the second component of our IE system: the sentential event recognizer.

4.3 Sentential Event Recognizer Models

The sentential event recognizer is one of the core contributions of this research, so in this section we evaluate it by itself, before we employ it within the unified framework. The purpose of the sentential event recognizer is to determine whether a sentence is discussing a domain-relevant event. For our data sets, the classifier must decide whether a sentence is discussing a terrorist incident (MUC-4) or a disease outbreak (ProMed). Ideally, we want such a classifier to operate independently from the answer keys and the extraction task per se. For example, a terrorism IE system could be designed to extract only perpetrators and victims of terrorist events, or it could be designed to extract only targets and locations. The job of the sentential event recognizer remains the same: to identify sentences that discuss a terrorist event. How to train and evaluate such a system is a difficult question. In this section, we present two approaches that we explored to generate the training data: (a) using the IE answer keys, and (b) using human judgements.

4.3.1 Sentence Annotation via Answer Keys

We have argued that the *event relevance* of a sentence should not be tied to a specific set of event roles. However, the IE answer keys can be used to identify some sentences that describe an event, because they contain an answer string. So we can map the answer strings back to sentences in the source documents to automatically generate event sentence annotations.[6] These annotations will be noisy, though, because an answer string can appear in a non-event sentence, and some event sentences may not contain any answer strings. The alternative, however, is sentence annotations by humans, which (as we will discuss in Section 4.3.2) is challenging.

4.3.2 Sentence Annotation via Human Judgements

For many sentences there is a clear consensus among people that an event is being discussed. For example, most readers would agree that sentence (1) below is describing a terrorist event, while sen-

[6] A similar strategy was used in previous work (Patwardhan and Riloff, 2007) to generate a test set for the evaluation of a relevant region classifier.

		Evaluation on Answer Keys							*Evaluation on Human Annotations*						
			Event			**Non-Event**				**Event**			**Non-Event**		
		Acc	Pr	Rec	F	Pr	Rec	F	Acc	Pr	Rec	F	Pr	Rec	F
MUC-4 (Terrorism)															
Ans	NB	.80	.57	.55	.56	.86	.87	.87	.81	.46	.60	.52	.91	.85	.88
	SVM	.80	.68	.42	.52	.84	.93	.88	.83	.55	.44	.49	.88	.91	.90
Hum	NB	.82	.64	.48	.55	.85	.92	.88	**.85**	.56	.57	.57	.91	.91	.91
	SVM	.79	.64	.41	.50	.83	.91	.87	.84	.62	.51	.56	.90	.91	.91
ProMed (Disease Outbreaks)															
Ans	NB	.75	.62	.61	.61	.81	.82	.82	.72	.43	.58	.50	.86	.77	.81
	SVM	.74	.78	.31	.44	.74	.95	.83	.76	.51	.26	.35	.80	.92	.86
Hum	NB	.73	.61	.46	.52	.77	.86	.81	**.79**	.56	.57	.56	.87	.86	.86
	SVM	.70	.62	.32	.42	.73	.89	.81	**.79**	.62	.42	.50	.84	.90	.87

Table 3: Sentential Event Recognizers Results (5-fold Cross-Validation)

	Evaluation on Human Annotations						
		Event			**Non-Event**		
	Acc	Pr	Rec	F	Pr	Rec	F
NB	.83	.50	.70	.58	.94	.86	.90
SVM	**.89**	.83	.39	.53	.89	.98	.94

Table 4: Sentential Event Recognizer Results for MUC-4 using 1300 Documents for Training

tence (2) is not. However it is difficult to draw a clear line. Sentence (3), for example, describes an action taken in response to a terrorist event. Is this a terrorist event sentence? Precisely how to define an *event sentence* is not obvious.

> (1) *Al Qaeda operatives launched an attack on the Madrid subway system.*
>
> (2) *Madrid has a population of about 3.2 million people.*
>
> (3) *City officials stepped up security in response to the attacks.*

We tackled this issue by creating detailed annotation guidelines to define the notion of an event sentence, and conducting a human annotation study. The guidelines delineated a general time frame for the beginning and end of an event, and constrained the task to focus on specific incidents that were reported in the IE answer key. We gave the annotators a brief description (e.g., *murder in Peru*) of each event that had a filled answer key in the data set. They only labeled sentences that discussed those particular events.

We employed two human judges, who annotated 120 documents from the ProMed test set, and 100 documents from the MUC-4 test set. We asked both judges to label 30 of the same documents from each data set so that we could compute inter-annotator agreement. The annotators had an agreement of 0.72 Cohen's κ on the ProMed data, and 0.77 Cohen's κ on the MUC-4 data. Given the difficulty of this task, we were satisfied that this task is reasonably well-defined and the annotations are of good quality.

4.3.3 Event Recognizer Results

We evaluated the two sentential event recognizer models described in Section 3.2 in two ways: (1) using the answer key sentence annotations for training/testing, and (2) using the human annotations for training/testing. Table 3 shows the results for all combinations of training/testing data. Since we only have human annotations for 100 MUC-4 texts and 120 ProMed texts, we performed 5-fold cross-validation on these documents. For our classifiers, we used the Weka (Witten and Frank, 2005) implementation of Naïve Bayes and the SVMLight (Joachims, 1998) implementation of the SVM. For each classifier we report overall accuracy, and precision, recall and F-scores with respect to both the positive and negative classes (event vs. non-event sentences).

The rows labeled *Ans* show the results for models trained via answer keys, and the rows labeled *Hum* show the results for the models trained with human annotations. The left side of the table shows the results using the answer key annotations for evaluation, and the right side of the table shows the results using the human annotations for evaluation. One expects classifiers to perform best when they are trained and tested on the same type of data, and our results bear this out – the classifiers that were trained and tested on the same kind of annotations do best. The boldfaced numbers represent the best accuracies achieved for each domain. As we would expect, the classifiers that are both trained and tested with human annotations (*Hum*) show the best performance, with the Naïve Bayes achieving the best accuracy of 85% on the

	PerpInd			PerpOrg			Target			Victim			Weapon		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
AutoSlog-TS	.33	.49	.40	.52	.33	.41	.54	.59	**.56**	.49	.54	.51	.38	.44	.41
Sem Affinity	.48	.39	.43	.36	.58	**.45**	.56	.46	.50	.46	.44	.45	.53	.46	.50
NB (baseline)	.36	.34	.35	.35	.46	.40	.53	.49	.51	.50	.50	.50	1.00	.05	.10
GLACIER															
NB/NB .90	.39	.59	.47	.33	.51	.40	.39	.72	.51	.52	.54	.53	.47	.55	.51
NB/SVM .40	.51	.58	.54	.34	.45	.38	.42	.72	.53	.55	.58	**.56**	.57	.53	**.55**
NB/SVM .50	.66	.47	**.55**	.41	.26	.32	.50	.62	.55	.62	.36	.45	.64	.43	.52

Table 5: Unified IE Model on MUC-4

MUC-4 texts, and the SVM achieving the best accuracy of 79% on the ProMed texts.

The recall and precision for non-event sentences is much higher than for event sentences. This classifier is forced to draw a hard line between the event and non-event sentences, which is a difficult task even for people. One of the advantages of our unified IE model, which will be described in the next section, is that it does not require hard decisions but instead uses a probabilistic estimate of how "event-ish" a sentence is.

Table 3 showed that models trained on human annotations outperform models trained on answer key annotations. But with the MUC-4 data, we have the luxury of 1300 training documents with answer keys, while we only have 100 documents with human annotations. Even though the answer key annotations are noisier, we have 13 times as much training data.

So we trained another sentential event recognizer using the entire MUC-4 training set. These results are shown in Table 4. Observe that using this larger (albeit noisy) training data does not appear to affect the Naïve Bayes model very much. Compared with the model trained on 100 manually annotated documents, its accuracy decreases by 2% from 85% to 83%. The SVM model, on the other hand, achieves an 89% accuracy when trained with the larger MUC-4 training data, compared to 84% accuracy for the model trained from the 100 manually labeled documents. Consequently, the sentential event recognizer models used in our unified IE framework (described in Section 4.4) are trained with this 1300 document training set.

4.4 Evaluation of the Unified IE Model

We now evaluate the performance of our unified IE model, GLACIER, which allows a plausible role-filler recognizer and a sentential event recognizer to make joint decisions about phrase extractions. Tables 5 and 6 present the results of the unified

	Disease			Victim		
	P	R	F	P	R	F
AutoSlog-TS	.33	.60	.43	.36	.49	.41
Sem Affinity	.31	.49	.38	.41	.47	**.44**
NB (baseline)	.34	.59	.43	.47	.39	.43
GLACIER						
NB/NB .90	.41	.61	**.49**	.38	.52	**.44**
NB/SVM .40	.31	.66	.42	.32	.55	.41
NB/SVM .50	.38	.54	.44	.42	.47	**.44**

Table 6: Unified IE Model on ProMed

IE model on the MUC-4 and ProMed data sets. The NB/NB systems use Naïve Bayes models for both components, while the NB/SVM systems use a Naïve Bayes model for the plausible role-filler recognizer and an SVM for the sentential event recognizer. As with our baseline system, we obtain good results using a threshold of 0.90 for our NB/NB model (i.e., only NPs with probability $\geq$ 0.90 are extracted). For our NB/SVM models, we evaluated using the default threshold (0.50) but observed that recall was sometimes low. So we also use a threshold of 0.40, which produces superior results. Here too, we used the Weka (Witten and Frank, 2005) implementation of the Naïve Bayes model and the SVMLight (Joachims, 1998) implementation of the SVM.

For the MUC-4 data, our unified IE model using the SVM (0.40) outperforms all 3 baselines on three roles (**PerpInd, Victim, Weapon**) and outperforms 2 of the 3 baselines on the **Target** role. When GLACIER outperforms the other systems it is often by a wide margin: the F-score for **PerpInd** jumped from 0.43 for the best baseline (Sem Affinity) to 0.54 for GLACIER, and the F-scores for **Victim** and **Weapon** each improved by 5% over the best baseline. These gains came from both increased recall and increased precision, demonstrating that GLACIER extracts some information that was missed by the other systems and is also less prone to false hits.

Only the **PerpOrg** role shows inferior performance. Organizations perpetrating a terrorist

event are often discussed later in a document, far removed from the main event description. For example, a statement that *Al Qaeda* is believed to be responsible for an attack would typically appear after the event description. As a result, the sentential event recognizer tends to generate low probabilities for such sentences. We believe that addressing this issue would require the use of discourse relations or the use of even larger context sizes. We intend to explore these avenues of research in future work.

On the ProMed data, GLACIER produces results that are similar to the baselines for the **Victim** role, but it outperforms the baselines for the **Disease** role. We find that for this domain, the unified IE model with the Naïve Bayes sentential event recognizer is superior to the unified IE model with the SVM classifier. For the **Disease** role, the F-score jumped 6%, from 0.43 for the best baseline systems (AutoSlog-TS and the NB baseline) to 0.49 for $\text{GLACIER}_{\text{NB/NB}}$. In contrast to the MUC-4 data, this improvement was mostly due to an increase in precision (up to 0.41), indicating that our unified IE model was effective at eliminating many false hits. For the **Victim** role, the performance of the unified model is comparable to the baselines. On this event role, the F-score of $\text{GLACIER}_{\text{NB/NB}}$ (0.44) matches that of the best baseline system (Sem Affinity, with 0.44). However, note that $\text{GLACIER}_{\text{NB/NB}}$ can achieve a 5% gain in recall over this baseline, at the cost of a 3% precision loss.

4.5 Specific Examples

Figure 2 presents some specific examples of extractions that are failed to be extracted by the baseline models, but are correctly identified by GLACIER because of its use of sentential evidence. Observe that in each of these examples, GLACIER correctly extracts the underlined phrases, in spite of the inconclusive evidence in the local contexts around them. In the last sentence in Figure 2, for example, GLACIER correctly makes the inference that the policemen in the bus (which was traveling on the bridge) are likely the victims of the terrorist event. Thus, we see that our system manages to balance the influence of the two probability components to make extraction decisions that would be impossible to make by relying only on the local phrasal context. In addition, the sentential event recognizer can also help improve precision by preventing extractions from non-event sentences.

THE MNR REPORTED ON 12 JANUARY THAT HEAVILY ARMED MEN IN CIVILIAN CLOTHES HAD INTERCEPTED A VEHICLE WITH OQUELI AND FLORES ENROUTE FOR LA AURORA AIRPORT AND THAT THE TWO POLITICAL LEADERS HAD BEEN KIDNAPPED AND WERE REPORTED MISSING.

PerpInd: HEAVILY ARMED MEN

THE SCANT POLICE INFORMATION SAID THAT THE DEVICES WERE APPARENTLY LEFT IN FRONT OF THE TWO BANK BRANCHES MINUTES BEFORE THE CURFEW BEGAN FOR THE 6TH CONSECUTIVE DAY – PRECISELY TO COUNTER THE WAVE OF TERRORISM CAUSED BY DRUG TRAFFICKERS.

Weapon: THE DEVICES

THOSE WOUNDED INCLUDE THREE EMPLOYEES OF THE GAS STATION WHERE THE CAR BOMB WENT OFF AND TWO PEOPLE WHO WERE WALKING BY THE GAS STATION AT THE MOMENT OF THE EXPLOSION.

Victim: THREE EMPLOYEES OF THE GAS STATION
Victim: TWO PEOPLE

MEMBERS OF THE BOMB SQUAD HAVE DEACTIVATED A POWERFUL BOMB PLANTED AT THE ANDRES AVELINO CACERES PARK, WHERE PRESIDENT ALAN GARCIA WAS DUE TO PARTICIPATE IN THE COMMEMORATION OF THE BATTLE OF TARAPACA.

Victim: PRESIDENT ALAN GARCIA

EPL [POPULAR LIBERATION ARMY] GUERRILLAS BLEW UP A BRIDGE AS A PUBLIC BUS, IN WHICH SEVERAL POLICEMEN WERE TRAVELING, WAS CROSSING IT.

Victim: SEVERAL POLICEMEN

Figure 2: Examples of GLACIER Extractions

5 Conclusions

We presented a unified model for IE that balances the influence of sentential context with local contextual evidence to improve the performance of event-based IE. Our experimental results showed that using sentential contexts indeed produced better results on two IE data sets. Our current model uses supervised learning, so one direction for future work is to adapt the model for weakly supervised learning. We also plan to incorporate discourse features and investigate even wider contexts to capture broader discourse effects.

Acknowledgments

This work has been supported in part by the Department of Homeland Security Grant N0014-07-1-0152. We are grateful to Nathan Gilbert and Adam Teichert for their help with the annotation of event sentences.

References

R. Bunescu and R. Mooney. 2004. Collective Information Extraction with Relational Markov Networks. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 438–445, Barcelona, Spain, July.

M. Califf and R. Mooney. 2003. Bottom-Up Relational Learning of Pattern Matching Rules for Information Extraction. *Journal of Machine Learning Research*, 4:177–210, December.

H. Chieu, H. Ng, and Y. Lee. 2003. Closing the Gap: Learning-Based Information Extraction Rivaling Knowledge-Engineering Methods. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 216–223, Sapporo, Japan, July.

P. Domingos and M. Pazzani. 1996. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 105–112, Bari, Italy, July.

J. Finkel, T. Grenager, and C. Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 363–370, Ann Arbor, MI, June.

D. Freitag and A. McCallum. 2000. Information Extraction with HMM Structures Learned by Stochastic Optimization. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 584–589, Austin, TX, August.

D. Freitag. 1998. Toward General-Purpose Learning for Information Extraction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 404–408, Montreal, Quebec, August.

H. Ji and R. Grishman. 2008. Refining Event Extraction through Cross-Document Inference. In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, OH, June.

T. Joachims. 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the Tenth European Conference on Machine Learning*, pages 137–142, April.

C. Manning, P. Raghavan, and H Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY.

M. Maslennikov and T. Chua. 2007. A Multi-resolution Framework for Information Extraction from Free Text. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 592–599, Prague, Czech Republic, June.

S. Patwardhan and E. Riloff. 2007. Effective Information Extraction with Semantic Affinity Patterns and Relevant Regions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 717–727, Prague, Czech Republic, June.

W. Phillips and E. Riloff. 2007. Exploiting Role-Identifying Nouns and Expressions for Information Extraction. In *Proceedings of International Conference on Recent Advances in Natural Language Processing*, pages 165–172, Borovets, Bulgaria, September.

E. Riloff and W. Phillips. 2004. An Introduction to the Sundance and AutoSlog Systems. Technical Report UUCS-04-015, School of Computing, University of Utah.

E. Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Articial Intelligence*, pages 1044–1049, Portland, OR, August.

S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. 1995. CRYSTAL: Inducing a Conceptual Dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1314–1319, Montreal, Canada, August.

B. Sundheim. 1992. Overview of the Fourth Message Understanding Evaluation and Conference. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages 3–21, McLean, VA, June.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer, New York, NY.

I. Witten and E. Frank. 2005. *Data Mining - Practical Machine Learning Tools and Techniques*. Morgan–Kaufmann, San Francisco, CA.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000. Automatic Acquisition of Domain Knowledge for Information Extraction. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 940–946, Saarbrücken, Germany, August.

B. Zadrozny and C. Elkan. 2001. Obtaining Calibrated Probability Estimates from Decision Trees and Naïve Bayesian Classiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 609–616, Williamstown, MA, June.

Review Sentiment Scoring via a Parse-and-Paraphrase Paradigm

Jingjing Liu, Stephanie Seneff
MIT Computer Science & Artificial Intelligence Laboratory
32 Vassar Street, Cambridge, MA 02139
{jingl, seneff}@csail.mit.edu

Abstract

This paper presents a parse-and-paraphrase paradigm to assess the degrees of sentiment for product reviews. Sentiment identification has been well studied; however, most previous work provides binary polarities only (positive and negative), and the polarity of sentiment is simply reversed when a negation is detected. The extraction of lexical features such as unigram/bigram also complicates the sentiment classification task, as linguistic structure such as implicit long-distance dependency is often disregarded. In this paper, we propose an approach to extracting adverb-adjective-noun phrases based on clause structure obtained by parsing sentences into a hierarchical representation. We also propose a robust general solution for modeling the contribution of adverbials and negation to the score for degree of sentiment. In an application involving extracting aspect-based pros and cons from restaurant reviews, we obtained a 45% relative improvement in recall through the use of parsing methods, while also improving precision.

1 Introduction

Online product reviews have provided an extensive collection of free-style texts as well as product ratings prepared by general users, which in return provide grassroots contributions to users interested in a particular product or service as assistance. Yet, valuable as they are, free-style reviews contain much noisy data and are tedious to read through in order to reach an overall conclusion. Thus, we conducted this study to automatically process and evaluate product reviews in order to generate both numerical evaluation and textual summaries of users' opinions, with the ultimate goal of adding value to real systems such as a restaurant-guide dialogue system.

Sentiment summarization has been well studied in the past decade (Turney, 2002; Pang et al., 2002; Dave et al., 2003; Hu and Liu, 2004a, 2004b; Carenini et al., 2006; Liu et al., 2007). The polarity of users' sentiments in each segment of review texts is extracted, and the polarities of individual sentiments are aggregated among all the sentences/segments of texts to give a numerical scaling on sentiment orientation.

Most of the work done for sentiment analysis so far has employed shallow parsing features such as part-of-speech tagging. Frequent adjectives and nouns/noun phrases are extracted as opinion words and representative product features. However, the linguistic structure of the sentence is usually not taken into consideration. High level linguistic features, if well utilized and accurately extracted, can provide much insight into the semantic meaning of user opinions and contribute to the task of sentiment identification.

Furthermore, in addition to adjectives and nouns, adverbials and negation also play an important role in determining the degree of the orientation level. For example, "very good" and "good" certainly express different degrees of positive sentiment. Also, in previous studies, when negative expressions are identified, the polarity of sentiment in the associated segment of text is simply reversed. However, semantic expressions are quite different from the absolute opposite values in mathematics. For example, "not bad" does not express the opposite meaning of "bad", which would be highly positive. Simply reversing the polarity of sentiment on the appearance of negations may result in inaccurate interpretation of sentiment expressions. Thus, a system which attempts to quantify sentiment while ignoring adverbials is missing a significant component of the sentiment score, especially if the adverbial is a negative word.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 161–169,
Singapore, 6-7 August 2009.

Another challenging aspect of negation is proper scoping of the negative reference over the right constituent, which we argue, can be handled quite well with careful linguistic analysis. Take the sentence "*I don't think the place is very clean*" as example. A linguistic approach associating long-distance elements with semantic relations can identify that the negation "not" scopes over the complement clause, thus extracting "not very clean" instead of "very clean".

Our goal in modeling adverbials is to investigate whether a simple linear correction model can capture the polarity contribution of all adverbials. Furthermore, is it also appropriate to adjust for multiple adverbs, including negation, via a linear *additive* model? I.e., can "not very good" be modeled as *not(very(good))*? The fact that "not very good" seems to be *less* negative than "not good" suggests that such an algorithm might work well. From these derivations we have developed a model which treats negations in the exact same way as modifying adverbs, via an accumulative linear offset model. This yields a very generic and straightforward solution to modeling the strength of sentiment expression.

In this paper we utilize a parse-and-paraphrase paradigm to identify semantically related phrases in review texts, taking quantifiers (e.g., modifying adverbs) and qualifiers (e.g., negations) into special consideration. The approach makes use of a lexicalized probabilistic syntactic grammar to identify and extract sets of *adverb-adjective-noun* phrases that match review-related patterns. Such patterns are constructed based on well-formed linguistic structure; thus, relevant phrases can be extracted reliably.

We also propose a cumulative linear offset model to calculate the degree of sentiment for joint adjectives and quantifiers/qualifiers. The proposed sentiment prediction model takes modifying adverbs and negations as universal scales on strength of sentiment, and conducts cumulative calculation on the degree of sentiment for the associated adjective. With this model, we can provide not only qualitative textual summarization such as "good food" and "bad service", but also a numerical scoring of sentiment, i.e., "how good the food is" and "how bad the service is."

2 Related Work

There have been many studies on sentiment classification and opinion summarization (Pang and Lee, 2004, 2005; Gamon et al., 2005; Popescu and Etzioni, 2005; Liu et al., 2005; Zhuang et al., 2006; Kim and Hovy, 2006). Specifically, aspect rating as an interesting topic has also been widely studied (Titov and McDonald, 2008a; Snyder and Barzilay, 2007; Goldberg and Zhu, 2006). Recently, Baccianella et. al. (2009) conducted a study on multi-facet rating of product reviews with special emphasis on how to generate vectorial representations of the text by means of POS tagging, sentiment analysis, and feature selection for ordinal regression learning. Titov and McDonald (2008b) proposed a joint model of text and aspect ratings which utilizes a modified LDA topic model to build topics that are representative of ratable aspects, and builds a set of sentiment predictors. Branavan et al. (2008) proposed a method for leveraging unstructured annotations in product reviews to infer semantic document properties, by clustering user annotations into semantic properties and tying the induced clusters to hidden topics in the text.

3 System Overview

Our review summarization task is to extract sets of descriptor-topic pairs (e.g., "excellent service") from a set of reviews (e.g., for a particular restaurant), and to cluster the extracted phrases into representative aspects on a set of dimensions (e.g., "food", "service" and "atmosphere"). Driven by this motivation, we propose a three-stage system that automatically processes reviews. A block diagram is given in Figure 1.

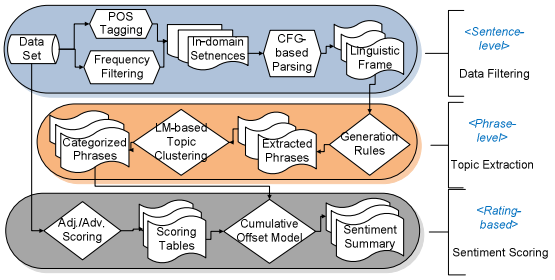


Figure 1. Framework of review processing.

The first stage is *sentence-level data filtering*. Review data published by general users is often in free-style, and a large fraction of the data is either ill-formed or not relevant to the task. We classify these as *out of domain* sentences. To filter out such noisy data, we collect unigram statistics on all the relevant words in the corpus, and select high frequency adjectives and nouns. Any sentence that contains none of the high-frequency nouns or adjectives is rejected from further analysis. The remaining *in-domain* sentences are subjected to the second stage, *parse*

analysis and semantic understanding, for topic extraction.

From the parsable sentences we extract descriptor-topic phrase patterns based on a carefully-designed generation grammar. We then apply *LM* (*language model) based topic clustering* to group the extracted phrases into representative aspects. The third stage scores the *degree of sentiment* for adjectives, as well as the *strength of sentiment* for modifying adverbs and negations, which further refine the degree of sentiment of the associated adjectives. We then run a *linear additive model* to assign a combined sentiment score for each extracted phrase.

The rest of the paper is structured as follows: In Section 4, we explain the linguistic analysis. In Section 5, we describe the cumulative model for assessing the degree of sentiment. Section 6 provides a systematic evaluation, conducted on real data in the restaurant review domain harvested from the Web. Section 7 provides a discussion analyzing the results. Section 8 summarizes the paper as well as pointing to future work.

4 Linguistic Analysis

4.1 Parse-and-Paraphrase

Our linguistic analysis is based on a parse-and-paraphrase paradigm. Instead of the flat structure of a surface string, the parser provides a hierarchical representation, which we call a *linguistic frame* (Xu et al., 2008). It preserves linguistic structure by encoding different layers of semantic dependencies. The grammar captures syntactic structure through a set of carefully constructed context free grammar rules, and employs a feature-passing mechanism to enforce long distance constraints. The grammar is lexicalized, and uses a statistical model to rank order competing hypotheses. It knows explicitly about 9,000 words, with all unknown words being interpreted as nouns. The grammar probability model was trained automatically on the corpus of review sentences.

To produce the phrases, a set of generation rules is carefully constructed to only extract sets of related adverbs, adjectives and nouns. The adjective-noun relationships are captured from the following linguistic patterns: (1) all adjectives attached directly to a noun in a noun phrase, (2) adjectives embedded in a relative clause modifying a noun, and (3) adjectives related to nouns in a subject-predicate relationship in a clause. These patterns are compatible, i.e., if a clause contains both a modifying adjective and a predicate adjective related to the same noun, two adjective-noun pairs are generated by different patterns. As in, "The efficient waitress was nonetheless very courteous." It is a "parse-and-paraphrase-like" paradigm: the paraphrase tries to preserve the original words intact, while reordering them and/or duplicating them into multiple NP units. Since they are based on syntactic structure, the generation rules can also be applied in any other domain involving opinion mining.

An example linguistic frame is shown in Figure 2, which encodes the sentence "*The caesar with salmon or chicken is really quite good.*" In this example, for the adjective "good", the nearby noun "chicken" would be associated with it if only proximity is considered. From the linguistic frame, however, we can easily associate "caesar" with "good" by extracting the head of the topic sub-frame and the head of the predicate sub-frame, which are encoded in the same layer (root layer) of the linguistic frame. Also, we can tell from the predicate sub-frame that there is an adverb "*quite*" modifying the head word "*good*". The linguistic frame also encodes an adverb "*really*" in the upstairs layer. A well-constructed generation grammar can create customized *adverb-adjective-noun* phrases such as "quite good caesar" or "really quite good caesar".

```
{c cstatement
 :topic {q caesar
         :quantifier "def"
         :pred {p with :topic {q salmon
                               :pred {p conjunction
                                      :or {q chicken }}}}
 :adv "really"
 :pred {p adj_complement
        :pred {p adjective
               :adv "quite"
               :pred {p quality :topic "good" }}}}
```

Figure 2. Linguistic frame for "*The caesar with salmon or chicken is really quite good.*"

Interpreting negation in English is not straightforward, and it is often impossible to do correctly without a deep linguistic analysis. Xuehui Wu (2005) wrote: "The scope of negation is a complex linguistic phenomenon. It is easy to perceive but hard to be defined from a syntactic point of view. Misunderstanding or ambiguity may occur when the negative scope is not understood clearly and correctly." The majority rule for negation is that it scopes over the remainder of its containing clause, and this works well for most cases. For example, Figure 3 shows

the linguistic frame for the sentence "*Their menu was a good one that didn't try to do too much.*"

```
{c cstatement
 :topic {q menu   :poss "their" } }
 :complement {q pronoun   :name "one"
        :adj_clause {c cstatement
                :conjn "that"
                :negate "not"
                :pred {p try :to_clause  {p do
                        :topic {q object
                        :adv "too"
                        :quant "much" }}}}
 :pred {p adjective
        :pred {p quality :topic "good" }}}
```

Figure 3. Linguistic frame for "*Their menu was a good one that didn't try to do too much.*"

Traditional approaches which do not consider the linguistic structure would treat the appearance of "*not*" as a negation and simply reverse the sentiment of the sentence to negative polarity, which is wrong as the sentence actually expresses positive opinion for the topic "menu". In our approach, the negation "*not*" is identified as under the sub-frame of the complement clause, instead of in the same or higher layer of the adjective sub-frame; thus it is considered as unrelated to the adjective "good". In this way we can successfully predict the scope of the reference of the negation over the correct constituent of a sentence and create proper association between negation and its modified words.

4.2 LM-based Topic Clustering

To categorize the extracted phrases into representative aspects, we automatically group the identified topics into a set of clusters based on LM probabilities. The LM-based algorithm assumes that topics which are semantically related have high probability of co-occurring with similar descriptive words. For example, "delicious" might co-occur frequently with both "pizza" and "dessert". By examining the distribution of bigram probability of these topics with corresponding descriptive words, we can group "pizza" and "dessert" into the same cluster of "food".

We select a small set of the most common topics, i.e., topics with the highest frequency counts, and put them into an initial set *I*. Then, for each candidate topic t_c outside set *I*, we calculate its probability given each topic t_i within the initial set *I*, given by:

$$P(t_c|\, t_i) = \sum_{a\in A} P(t_c|a) \cdot P(a|t_i)$$
$$= \sum_{a\in A} \frac{P(a,t_c)}{P(a)} \cdot \frac{P(a,t_i)}{P(t_i)}$$
$$= \frac{1}{P(t_i)} \sum_{a\in A} \frac{1}{P(a)} \cdot P(a,t_c) \cdot P(a,t_i) \quad (1)$$

where *A* represents the set of all the adjectives in the corpus. For each candidate topic t_c, we choose the cluster of the initial topic t_i with which it has the highest probability score.

There are also cases where a meaningful adjective occurs in the absence of an associated topic, e.g., "It is quite *expensive*." We call such cases the "*widow-adjective*" case. Without hard-coded ontology matching, it is difficult to identify "expensive" as a price-related expression. To discover such cases and associate them with related topics, we propose a "*surrogate topic*" matching approach based on bigram probability.

As aforementioned, the linguistic frame organizes all adjectives into separate clauses. Thus, we create a "surrogate topic" category in the linguistic frames for widow-adjective cases, which makes it easy to detect descriptors that are affiliated with uninformative topics like the pronoun "it". We then have it generate phrases such as "expensive surrogate_topic" and use bigram probability statistics to automatically map each sufficiently strongly associated adjective to its most common topic among our major classes, e.g., mapping "expensive" with its surrogate topic "price". Therefore, we can generate sets of additional phrases in which the topic is "hallucinated" from the widow-adjective.

5 Assessment of Sentiment Strength

5.1 Problem Formulation

Given the sets of adverb-adjective-noun phrases extracted by linguistic analysis, our goal is to assign a score for the degree of sentiment to each phrase and calculate an average rating for each aspect. An example summary is given in Table 1.

Table 1. Example of review summary.

Aspect	Extracted phrases	Rating
Atmosphere	very nice ambiance, outdoor patio	4.8
Food	not bad meal, quite authentic food	4.1
Place	not great place, very smoky restaurant	2.8
Price	so high bill, high cost, not cheap price	2.2

To calculate the numerical degree of sentiment, there are three major problems to solve: 1) how to associate *numerical scores* with *textual sentiment*; 2) whether to calculate sentiment scores for adjectives and adverbs *jointly* or *separately*; 3)

whether to treat negations as *special cases* or in the *same way* as modifying adverbs.

There have been studies on building sentiment lexicons to define the strength of sentiment of words. Esuli and Sebastiani (2006) constructed a lexical resource, SentiWordNet, a WordNet-like lexicon emphasizing sentiment orientation of words and providing numerical scores of how objective, positive and negative these words are. However, lexicon-based methods can be tedious and inefficient and may not be accurate due to the complex cross-relations in dictionaries like WordNet. Instead, our primary approach to sentiment scoring is to make use of collective data such as user ratings. In product reviews collected from online forums, the format of a review entry often consists of three parts: pros/cons, free-style text and user rating. We assume that user rating is normally consistent with the tone of the review text published by the same user. By associating user ratings with each phrase extracted from review texts, we can easily associate numerical scores with textual sentiment.

A simple strategy of rating assignment is to take each extracted adverb-adjective pair as a composite unit. However, this method is likely to lead to a large number of rare combinations, thus suffering from sparse data problems. Therefore, an interesting question to ask is whether it is feasible to assign to each adverb a perturbation score, which adjusts the score of the associated adjective up or down by a fixed scalar value. This approach thus hypothesizes that "very expensive" is as much worse than "expensive" as "very romantic" is better than "romantic". This allows us to pool all instances of a given adverb regardless of which adjective it is associated with, in order to compute the absolute value of the perturbation score for that adverb. Therefore, we consider adverbs and adjectives separately when calculating the sentiment score, treating each modifying adverb as a universal quantifier which consistently scales up/down the strength of sentiment for the adjectives it modifies.

Furthermore, instead of treating negation as a special case, the universal model works for all adverbials. The model hypothesizes that "not bad" is as much better than "bad" as "not good" is worse than "good", i.e., negations push positive/negative adjectives to the other side of sentiment polarity by a *universal scale*. This again, allows us to pool all instances of a given negation and compute the absolute value of the perturbation score for that negation, in the same way as dealing with modifying adverbs.

5.2 Linear Additive Model

For each adjective, we average all its ratings given by:

$$Score(adj) = \frac{\sum_{i \in P} \frac{N}{n_{r_i}} \cdot r_i}{\sum_{r_i} \frac{N}{n_{r_i}}} \qquad (2)$$

where P represents the set of appearances of adjective adj, r_i represents the associated user rating in each appearance of adj, N represents the number of entities (e.g., restaurants) in the entire data set, and n_{r_i} represents the number of entities with rating r_i. The score is averaged over all the appearances, weighted by the frequency count of each category of rating to remove bias towards any category.

As for adverbs, using a slightly modified version of equation (2), we can get a rating table for all *adverb-adjective* pairs. For each adverb *adv*, we get a list of all its possible combinations with adjectives. Then, for each *adj* in the list, we calculate the distance between the rating of *adv-adj* and the rating of the *adj* alone. We then aggregate the distances among all the pairs of *adv-adj* and *adj* in the list, weighted by the frequency count of each *adv-adj* pair:

$$Score(adv) = \sum_{t \in A} \frac{count(adv, adj_t)}{\sum_{j \in A} count(adv, adj_j)} \cdot Pol(adj_t) \cdot (r(adv, adj_t) - r(adj_t)) \qquad (3)$$

where $count(adv,\ adj_t)$ represents the count of the combination $adv - adj_t$, A represents the set of adjectives that co-occur with adv, $r(adv, adj_t)$ represents the sentiment rating of the combination $adv - adj_t$, and $r(adj_t)$ represents the sentiment rating of the adjective adj_t alone. $Pol(adj_t)$ represents the polarity of adj_t, assigned as *1* if adj_t is positive, and *-1* if negative.

Specifically, negations are well handled by the same scoring strategy, treated exactly the same as modifying adverbs, except that they get such strong negative scores that the sentiment of the associated adjectives is pushed to the other side of the polarity scale.

After obtaining the strength rating for adverbs and the sentiment rating for adjectives, the next step is to assign the strength of sentiment to each phrase (negation-adverb-adjective-noun) extracted by linguistic analysis, as given by:

$$Score\left(neg\left(adv(adj)\right)\right) = r(adj) + Pol(adj) \cdot r(adv) + Pol(adj) \cdot r(neg) \qquad (4)$$

where $r(adj)$ represents the rating of adjective adj, $r(adv)$ represents the rating of adverb adv, and $r(neg)$ represents the rating of negation neg. $Pol(adj)$ represents the polarity of adj, assigned as *1* if adj is positive, and *-1* if negative. Thus, if adj is positive, we assign a combined rating $r(adj) + r(adv)$ to this phrase. If it is negative, we assign $r(adj) - r(adv)$. Specifically, if it is a negation case, we further assign a linear offset $r(neg)$ if adj is positive or $-r(neg)$ if adj is negative. For example, given the ratings <good: 4.5>, <bad: 1.5>, <very: 0.5> and <not: -3.0>, we would assign "5.0" to "very good" (*score(very(good))=4.5+0.5*), "1.0" to "very bad" (*score(very(bad))=1.5-0.5*), and "2.0" to "not very good" (s*core(not(very(good)))= 4.5+0.5-3.0*). The corresponding sequence of different degrees of sentiment is: "*very good*: 5.0" > "*good*: 4.5" > "*not very good*: 2.0" > "*bad*: 1.5" > "*very bad*: 1.0".

6 Experiments

In this section we present a systematic evaluation of the proposed approaches conducted on real data. We crawled a data collection of 137,569 reviews on 24,043 restaurants in 9 cities in the U.S. from an online restaurant evaluation website[1]. Most of the reviews have both pros/cons and free-style text. For the purpose of evaluation, we take those reviews containing pros/cons as the experimental set, which is 72.7% (99,147 reviews) of the original set.

6.1 Topic Extraction

Based on the experimental set, we first filtered out-of-domain sentences based on frequency count, leaving a set of 857,466 in-domain sentences (67.5%). This set was then subjected to parse analysis; 78.6% of them are parsable.

Given the parsing results in the format of linguistic frame, we used a set of language generation rules to extract relevant adverb-adjective-noun phrases. We then selected the most frequent 6 topics that represented appropriate dimensions for the restaurant domain ("place", "food", "service", "price", "atmosphere" and "portion") as the initial set, and clustered the extracted topic mentions into different aspect categories by creating a set of topic mappings with the LM-based clustering method. Phrases not belonging to any category are filtered out.

[1] http://www.citysearch.com

To evaluate the performance of the proposed approach (LING) to topic extraction, we compare it with a baseline method similar to (Hu and Liu, 2004a, 2004b; Liu et al., 2005). We performed part-of-speech tagging on both parsable and unparsable sentences, extracted each pair of noun and adjective that has the smallest proximity, and filtered out those with low frequency counts. Adverbs and negation words that are adjacent to the identified adjectives were also extracted along with the adjective-noun pairs. We call this the "neighbor baseline" (NB).

The proposed method is unable to make use of the non-parsable sentences, which make up over 20% of the data. Hence, it seems plausible to utilize a back-off mechanism for these sentences via a combined system (COMB) incorporating NB only for the sentences that fail to parse.

In considering how to construct the "ground truth" set of pros and cons for particular aspects, our goal was to minimize error as much as possible without requiring exorbitant amounts of manual labeling. We also wanted to assure that the methods were equally fair to both systems (LING and NB). To these ends, we decided to pool together all of the topic mappings and surrogate topic hallucinations obtained automatically from both systems, and then to manually edit the resulting list to eliminate any that were deemed unreasonable. We then applied these edited mappings in an automatic procedure to the adjective-noun pairs in the user-provided pros and cons of all the restaurant reviews. The resulting aspect-categorized phrase lists are taken as the ground truth. Each system then used its own (unedited) set of mappings in processing the associated review texts.

We also needed an algorithm to decide on a particular set of reviews for consideration, again, with the goal of omitting bias towards either of the two systems. We decided to retain as the evaluation set all reviews which obtained at least one topic extraction from both systems. Thus the two systems processed exactly the same data with exactly the same definitions of "ground truth". Performance was evaluated on this set of 62,588 reviews in terms of recall (percentage of topics in the ground truth that are also identified from the review body) and precision (percentage of extracted topics that are also in the ground truth). These measures are computed separately for each review, and then averaged over all reviews.

As shown in Table 2, without clustering, the LING approach gets 4.6% higher recall than the

NB baseline. And the recall from the COMB approach is 3.9% higher than that from the LING approach and 8.5% higher than that from the NB baseline. With topic clustering, the COMB approach also gets the highest recall, with a 4.9% and 17.5% increase from the LING approach and the NB baseline respectively. The precision is quite close among the different approaches, around 60%. Table 2 also shows that the topic clustering approach increases the recall by 4.8% for the NB baseline, 12.8% for the LING approach, and 13.8% for the COMB approach.

Table 2. Experimental results of topic extraction by the NB baseline, the proposed LING approach and a combined system (COMB).

	No Clustering		
	NB	**LING**	**COMB**
Recall	39.6%	44.2%	48.1%
Precision	60.2%	60.0%	59.8%
	With Clustering		
	NB	**LING**	**COMB**
Recall	44.4%	57.0%	61.9%
Precision	56.8%	61.1%	60.8%

6.2 Sentiment Scoring

To score the degree of sentiment for each extracted phrase, we built a table of sentiment score (<*adjective*: *score*>) for adjectives and a table of strength score (<*adverb*: *score*>) for adverbs. The pros/cons often contain short and well-structured phrases, and have better parsing quality than the long and complex sentences in free-style texts; pros/cons also have clear sentiment orientations. Thus, we use pros/cons to score the sentiment of adjectives, which requires strong polarity association. To obtain reliable ratings, we associate the adjectives in the "*pros*" of review entries which have a user rating 4 or 5, and associate the adjectives in the "*cons*" of review entries with user ratings 1 or 2 (the scale of user rating is 1 to 5). Reviews with rating 3 are on the boundary of sentiment, so we associate both sides with the overall rating. On the other hand, the frequencies of adverbs in free-style texts are much higher than those in pros/cons, as pros/cons mostly contain *adjective-noun* patterns. Thus, we use free-style texts instead of pros/cons to score the strength of adverbs.

Partial results of the sentiment scoring are shown in Tables 3 and 4. As shown in Table 3, the polarity of sentiment as well as the degree of polarity of an adjective can be distinguished by its score. The higher the sentiment score is, the more positive the adjective is.

Table 3. Sentiment scoring for selected adjectives.

Adjective	**Rating**	**Adjective**	**Rating**
Excellent	5.0	Awesome	4.8
Easy	4.1	Great	4.4
Good	3.9	Limited	3.4
Inattentive	2.75	Overpriced	2.3
Rude	1.69	Horrible	1.3

Table 4 gives the scores of strength for most common adverbs. The higher the strength score is, the more the adverb scales up/down the degree of sentiment of the adjective it modifies. While "not" gets a strong negative score, some adverbs such as "a little" (-0.65) and "a bit" (-0.83) also get negative scores, indicating slightly less sentiment for the associated adjectives.

Table 4. Strength scoring for selected adverbs.

Adverb	**Rating**	**Adverb**	**Rating**
Super	0.58	Fairly	0.13
Extremely	0.54	Pretty	0.07
Incredibly	0.49	A little	-0.65
Very	0.44	A bit	-0.83
Really	0.39	Not	-3.10

To evaluate the performance of sentiment scoring, we randomly selected a subset of 1,000 adjective-noun phrases and asked two annotators to independently rate the sentiment of each phrase on a scale of 1 to 5. We compared the sentiment scoring between our system and the annotations in a measurement of mean distance:

$$distance = \frac{1}{|S|}\sum_{p \in S} |r_{ip} - r_{ap}| \quad (5)$$

where S represents the set of phrases, p represents each phrase in the set S, r_{ip} represents the rating on phrase p from our sentiment scoring system, and r_{ap} represents the annotated rating on phrase p. As shown in Table 5, the obtained mean distance between the scoring from our approach and that from each annotation set is 0.46 and 0.43 respectively, based on the absolute rating scale from 1 to 5. This shows that the scoring of sentiment from our system is quite close to human annotation. The kappa agreement between the two annotation sets is 0.68, indicating high consistency between the annotators. The reliability of these results gives us sufficient confidence to make use of the scores of sentiments for summarization.

To examine the prediction of sentiment polarity, for each annotation set, we pooled the phrases with rating 4/5 into "positive", rating 1/2 into "negative", and rating 3 into "neutral". Then we rounded up the sentiment scores from our system to integers and pooled the scores into three polar-

ity sets ("positive", "negative" and "neutral") in the same way. As shown in Table 5, the obtained kappa agreement between the result from our system and that from each annotation set is 0.55 and 0.60 respectively. This shows reasonably high agreement on the polarity of sentiment between our system and human evaluation.

Table 5. Comparison of sentiment scoring between the proposed approach and two annotation sets.

	Annotation 1	**Annotation 2**
Mean distance	0.46	0.43
Kappa agreement	0.55	0.60

Table 6. Experimental results of topic extraction based on sentiment polarity matching.

	No Clustering		
	NB	**LING**	**COMB**
Recall	34.5%	38.9%	42.2%
Precision	53.8%	54.0%	53.3%
	With Clustering		
	NB	**LING**	**COMB**
Recall	37.4%	49.7%	54.1%
Precision	48.5%	52.9%	51.4%

To evaluate the combination of topic extraction and sentiment identification, we repeated the topic extraction experiments presented in Table 2, but this time requiring as well a correct polarity assignment to obtain a match with the pros/cons ground truth. As shown in Table 6, the COMB approach gets the highest recall both with and without topic clustering, and the recall from the LING approach is higher than that from the NB baseline in both cases as well, indicating the superiority of the proposed approach. The precision is stable among the different approaches, consistent with the case without the consideration of sentiment polarity.

7 Discussion

It is surprising that the parse-and-paraphrase method performs so well, despite the fact that it utilizes less than 80% of the data (parsable set). In this section, we will discuss two experiments that were done to tease apart the contributions of different variables. In both experiments, we compared the change in relative improvement in recall between NB and LING, relative to the values in Table 6, in the with-clustering condition. In the table, LING obtains a score of 49.7% for recall, which is a 33% relative increase from the score for NB (37.4%). Three distinct factors could play a role in the improvement: the widow-adjective topic hallucinations, the topic mapping for clustering, and the extracted phrases themselves. An experiment involving omitting topic hallucinations from widow adjectives determined that these account for 12% of the relative increase. To evaluate the contribution of clustering, we replaced the mapping tables used by both systems with the edited one used by the ground truth computation. Thus, both systems made use of the same mapping table, removing this variable from consideration. This improved the performance of both systems (NB and LING), but resulted in a decrease of LING's relative improvement by 17%. This implies that LING's mapping table is superior. Since both systems use the same sentiment scores for adjectives and adverbs, the remainder of the difference (71%) must be due simply to higher quality extracted phrases.

We suspected that over-generated phrases (the 40% of phrases that find no mappings in the pros/cons) might not really be a problem. To test this hypothesis, we selected 100 reviews for their high density of extracted phrases, and manually evaluated all the over-generated phrases. We found that over 80% were well formed, correct, and informative. Therefore, a lower precision here does not necessarily mean poor performance, but instead shows that the pros/cons provided by users are often incomplete. By extracting summaries from review texts we can recover additional valuable information.

8 Conclusions & Future Work

This paper presents a parse-and-paraphrase approach to assessing the degree of sentiment for product reviews. A general purpose context free grammar is employed to parse review sentences, and semantic understanding methods are developed to extract representative negation-adverb-adjective-noun phrases based on well-defined semantic rules. A language modeling-based method is proposed to cluster topics into respective categories. We also introduced in this paper a cumulative linear offset model for supporting the assessment of the strength of sentiment in adjectives and quantifiers/qualifiers (including negations) on a numerical scale. We demonstrated that the parse-and-paraphrase method can perform substantially better than a neighbor baseline on topic extraction from reviews even with less data. The future work focuses in two directions: (1) building a relational database from the summaries and ratings and using it to enhance users' experiences in a multimodal spoken dialogue system; and (2) applying our techniques to other domains to demonstrate generality.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2009. Multi-facet Rating of Product Reviews. In Proceedings of European Conference on Information Retrieval.

S.R.K. Branavan, Harr Chen, Jacob Eisenstein, and Regina Barzilay. 2008. Learning document-level semantic properties from free-text annotations. In Proceedings of the Annual Conference of the Association for Computational Linguistics.

Giuseppe Carenini, Raymond Ng, and Adam Pauls 2006. Multi-Document Summarization of Evaluative Text. In Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics.

Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In Proceedings of the International Conference on World Wide Web.

Andrea Esuli and Fabrizio Sebastiani. 2006. SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining. In Proceedings of the 5th Conference on Language Resources and Evaluation.

Michael Gamon, Anthony Aue, Simon Corston-Oliver, and Eric Ringger . 2005. Pulse: Mining customer opinions from free text. In Proceedings of the 6th International Symposium on Intelligent Data Analysis.

Andrew Goldberg and Xiaojin Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In HLT-NAACL 2006 Workshop on Textgraphs: Graph-based Algorithms for Natural Language Processing.

Minqing Hu and Bing Liu. 2004a. Mining and summarizing customer reviews. In Proceedings of the 2004 ACM SIGKDD international conference on Knowledge Discovery and Data mining.

Minqing Hu and Bing Liu. 2004b. Mining Opinion Features in Customer Reviews. In Proceedings of Nineteenth National Conference on Artificial Intelligence.

Soo-Min Kim and Eduard Hovy. 2006. Automatic identification of pro and con reasons in online reviews. In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 483–490.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In Proceedings of International Conference on World Wide Web.

Jingjing Liu, Yunbo Cao, Chin-Yew Lin, Yalou Huang, and Ming Zhou. 2007. Low-Quality Product Review Detection in Opinion Summarization. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In Proceedings of the Annual Conference of the Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Proceedings of the Annual Conference of the Association for Computational Linguistics.

Bo Pang, Lillian Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.

A.M. Popescu and O. Etzioni. 2005. Extracting product features and opinions from reviews. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.

Benjamin Snyder and Regina Barzilay. 2007. Multiple Aspect Ranking using the Good Grief Algorithm. In Proceedings of the Joint Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies.

Ivan Titov and Ryan McDonald. 2008a. Modeling online reviews with multi-grain topic models. In Proceedings of the 17h International Conference on World Wide Web.

Ivan Titov and Ryan McDonald. 2008b. A Joint Model of Text and Aspect Ratings for Sentiment Summarization. In Proceedings of the Annual Conference of the Association for Computational Linguistics.

Peter D. Turney. 2002. Thumbs up or thumbs down? Sentiment orientation applied to unsupervised classification of reviews. In Proceedings of the Annual Conference of the Association for Computational Linguistics.

Xuehui Wu, 2005. On the Scope of Negation in English, Sino-US English Teaching, Vol. 2, No. 9, Sep. 2005. pp. 53-56.

Yushi Xu, Jingjing Liu, Stephanie Seneff. 2008. Mandarin Language Understanding in Dialogue Context. In Proceedings of International Symposium on Chinese Spoken Language Processing.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In Proceedings of the 15th ACM international conference on Information and knowledge management.

Supervised and Unsupervised Methods in Employing Discourse Relations for Improving Opinion Polarity Classification

Swapna Somasundaran Univ. of Pittsburgh, Pittsburgh, PA 15260, swapna@cs.pitt.edu
Galileo Namata Univ. of Maryland, College Park, MD 20742, namatag@cs.umd.edu
Janyce Wiebe Univ. of Pittsburgh, Pittsburgh, PA 15260, wiebe@cs.pitt.edu
Lise Getoor Univ. of Maryland, College Park, MD 20742, getoor@cs.umd.edu

Abstract

This work investigates design choices in modeling a discourse scheme for improving opinion polarity classification. For this, two diverse global inference paradigms are used: a supervised collective classification framework and an unsupervised optimization framework. Both approaches perform substantially better than baseline approaches, establishing the efficacy of the methods and the underlying discourse scheme. We also present quantitative and qualitative analyses showing how the improvements are achieved.

1 Introduction

The importance of discourse in opinion analysis is being increasingly recognized (Polanyi and Zaenen, 2006). Motivated by the need to enable discourse-based opinion analysis, previous research (Asher et al., 2008; Somasundaran et al., 2008) developed discourse schemes and created manually annotated corpora. However, it was not known whether and how well these linguistic ideas and schemes can be translated into effective computational implementations.

In this paper, we first investigate ways in which an opinion discourse scheme can be computationally modeled, and then how it can be utilized to improve polarity classification. Specifically, the discourse scheme we use is from Somasundaran et al. (2008), which was developed to support a global, interdependent polarity interpretation. To achieve discourse-based global inference, we explore two different frameworks. The first is a supervised framework that learns interdependent opinion interpretations from training data. The second is an unsupervised optimization framework which uses constraints to express the ideas of coherent opinion interpretation embodied in the scheme. For the supervised framework, we use Iterative Collective Classification (ICA), which facilitates machine learning using relational information. The unsupervised optimization is implemented as an Integer Linear Programming (ILP) problem. Via our implementations, we aim to empirically test if discourse-based approaches to opinion analysis are useful.

Our results show that both of our implementations achieve significantly better accuracies in polarity classification than classifiers using local information alone. This confirms the hypothesis that the discourse-based scheme is useful, and also shows that both of our design choices are effective. We also find that there is a difference in the way ICA and ILP achieve improvements, and a simple hybrid approach, which incorporates the strengths of both, is able to achieve significant overall improvements over both. Our analyses show that even when our discourse-based methods bootstrap from noisy classifications, they can achieve good improvements.

The rest of this paper is organized as follows: we discuss related work in Section 2 and the discourse scheme in Section 3. We present our discourse-based implementations in Section 4, experiments in Section 5, discussions in Section 6 and conclusions in Section 7.

2 Related Work

Previous work on polarity disambiguation has used contextual clues and reversal words (Wilson et al., 2005; Kennedy and Inkpen, 2006; Kanayama and Nasukawa, 2006; Devitt and Ahmad, 2007; Sadamitsu et al., 2008). However, these do not capture discourse-level relations.

Researchers, such as (Polanyi and Zaenen, 2006), have discussed how the discourse structure can influence opinion interpretation; and previous work, such as (Asher et al., 2008; Somasundaran et al., 2008), have developed annota-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 170–179,
Singapore, 6-7 August 2009.

tion schemes for interpreting opinions with discourse relations. However, they do not empirically demonstrate how automatic methods can use their ideas to improve polarity classification. In this work, we demonstrate concrete ways in which a discourse-based scheme can be modeled using global inference paradigms.

Joint models have been previously explored for other NLP problems (Haghighi et al., 2005; Moschitti et al., 2006; Moschitti, 2009). Our global inference model focuses on opinion polarity recognition task.

The biggest difference between this work and previous work in opinion analysis that use global inference methods is in the type of linguistic relations used to achieve the global inference. Some of the work is not related to discourse at all (e.g., lexical similarities (Takamura et al., 2007), morphosyntactic similarities (Popescu and Etzioni, 2005) and word-based measures like TF-IDF (Goldberg and Zhu, 2006)). Others use sentence cohesion (Pang and Lee, 2004), agreement/disagreement between speakers (Thomas et al., 2006; Bansal et al., 2008), or structural adjacency. In contrast, our work focuses on discourse-based relations for global inference. Another difference from the above work is that our work is over multi-party conversations.

Previous work on emotion and subjectivity detection in multi-party conversations has explored using prosodic information (Neiberg et al., 2006), combining linguistic and acoustic information (Raaijmakers et al., 2008) and combining lexical and dialog information (Somasundaran et al., 2007). Our work is focused on harnessing discourse-based knowledge and on interdependent inference.

There are several collective classification frameworks, including (Neville and Jensen, 2000; Lu and Getoor, 2003; Taskar et al., 2004; Richardson and Domingos, 2006; Bilgic et al., 2007). In this paper, we use an approach by (Lu and Getoor, 2003) which iteratively predicts class values using local and relational features. ILP has been used on other NLP tasks, e.g., (Denis and Baldridge, 2007; Choi et al., 2006; Roth and Yih, 2004). In this work, we employ ILP for modeling discourse constraints for polarity classification.

3 Discourse Scheme and Data

The scheme in Somasundaran et al. (2008) has been developed and annotated over the AMI meeting corpus (Carletta et al., 2005).[1] This scheme annotates opinions, their polarities (positive, negative, neutral) and their targets (a target is what the opinion is about). The targets of opinions are related via two types of relations: the *same* relation, which relates targets referring to the same entity or proposition, and the *alternative* relation, which relates targets referring to mutually exclusive options in the context of the discourse. Additionally, the scheme relates opinions via two types of *frame* relations: the *reinforcing* and *non-reinforcing* relations. The frame relations represent discourse scenarios: reinforcing relations exist between opinions when they contribute to the same overall stance, while non-reinforcing relations exist between opinions that show ambivalence.

The opinion annotations are text-span based, while in this work, we use Dialog Act (DA) based segmentation of meetings.[2] As the DAs are our units of classification, we map opinion annotations to the DA units as follows. If a DA unit contains an opinion annotation, the label is transferred upwards to the containing DA. When a DA contains multiple opinion annotations, each with a different polarity, one of them is randomly chosen as the label for the DA. The discourse relations existing between opinions are also transferred upwards, between the DAs containing each of these annotations. We recreate an example from Somasundaran et al. (2008) using DA segmentation in Example 1. Here, the speaker has a positive opinion towards the rubbery material for the TV remote.

(1) DA-1: ... this kind of rubbery material,
DA-2: *it's* a **bit more bouncy**,
DA-3: like you said they get chucked around a lot.
DA-4: A **bit more durable** and *that* can also be **ergonomic** and
DA-5: *it* kind of feels **a bit different from all the other remote controls**.

In the example, the individual opinion expressions (shown in bold) are essentially regarding the same thing – the rubbery material. Thus, the explicit targets (shown in *italics*), *it's*, *that*, and *it*, and the implicit target of **a bit more durable** are all linked

[1] The AMI corpus contains a set of scenario-based meetings where participants have to design a new TV remote prototype.

[2] DA segmentation is provided with the AMI corpus.

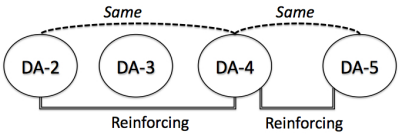


Figure 1: Discourse Relations between DA segments for Example 1.

with *same* target relations. Also, notice that the opinions reinforce a particular stance, i.e., a pro-rubbery-material stance. Thus, the scheme links the opinions via reinforcing relations. Figure 1 illustrates the corresponding discourse relations between the containing DA units.

4 Implementing the Discourse Model

The hypothesis in using discourse information for polarity classification is that the global discourse view will improve upon a classification with only a local view. Thus, we implement a local classifier to bootstrap the classification process, and then implement classifiers that use discourse information from the scheme annotations, over it. We explore two approaches for implementing our discourse-based classifier. The first is ICA, where discourse relations and the neighborhood information brought in by these relations are incorporated as features into the learner. The second approach is ILP optimization, which tries to maximize the class distributions predicted by the local classifier, subject to constraints imposed by discourse relations. Both classifiers thus accommodate preferences of the local classifier and for coherence with discourse neighbors.

4.1 Local Classifier

A supervised local classifier, *Local*, is used to provide the classifications to bootstrap the discourse-based classifiers.[3] It is important to make Local as reliable as possible; otherwise, the discourse relations will propagate misclassifications. Thus, we build Local using a variety of knowledge sources that have been shown to be useful for opinion analysis in previous work. Specifically, we construct features using polarity lexicons (used by (Wilson et al., 2005)), DA tags (used by (Somasundaran et al., 2007)) and unigrams (used by many researchers, e.g., (Pang and Lee, 2004)).

Note that, as our discourse-based classifiers attempt to improve upon the local classifications, Local is also a baseline for our experiments.

4.2 Iterative Collective Classification

We use a variant of ICA (Lu and Getoor, 2003; Neville and Jensen, 2000), which is a collective classification algorithm shown to perform consistently well over a wide variety of relational data.

Algorithm 1 ICA Algorithm

```
for each instance i do {bootstrapping}
   Compute polarity for i using local attributes
end for
repeat {iterative}
   Generate ordering I over all instances
   for each i in I do
      Compute polarity for i using local and relational attributes
   end for
until Stopping criterion is met
```

ICA uses two classifiers: a local classifier and a *relational classifier*. The local classifier is trained to predict the DA labels using only the local features. We use Local, described in Section 4.1, for this purpose. The relational classifier is trained using the local features, and an additional set of features commonly referred to as *relational features*. The value of a relational feature, for a given DA, depends on the polarity of the discourse neighbors of that DA. Thus, the relational features incorporate discourse and neighbor information; that is, they incorporate the information about the frame and target relations in conjunction with the polarity of the discourse neighbors. Intuitively, our motivation for this approach can be explained using Example 1. Here, in interpreting the ambiguous opinion **a bit different** as being positive, we use the knowledge that it participates in a reinforcing discourse, and that all its neighbors (e.g., **ergonomic**, **durable**) are positive opinions regarding the same thing. On the other hand, if it had been a non-reinforcing discourse, then the polarity of **a bit different**, when viewed with respect to the other opinions, could have been interpreted as negative.

Table 1 lists the relational features we defined for our experiments where each row represents a

[3]Local is supervised, as previous work has shown that supervised methods are effective in opinion analysis. Even though this makes the final end-to-end system with the ILP implementation semi-supervised, note that the discourse-based ILP part is itself unsupervised.

Percent of neighbors with polarity type a related via frame relation f'
Percent of neighbors with polarity type a related via target relation t'
Percent of neighbors with polarity type a related via frame relation f and target relation t
Percent of neighbors with polarity type a and same speaker related via frame relation f'
Percent of neighbors with polarity type a and same speaker related via target relation t'
Percent of neighbors with polarity type a related via a frame relation or target relation
Percent of neighbors with polarity type a related via a reinforcing frame relation or *same* target relation
Percent of neighbors with polarity type a related via a non-reinforcing frame relation or alt target relation
Most common polarity type of neighbors related via a *same* target relation
Most common polarity type of neighbors related via a reinforcing frame relation and *same* target relation

Table 1: Relational features: $a \in$ {non-neutral (i.e., positive or negative), positive, negative}, $t \in$ {same, alt}, $f \in$ {reinforcing, non-reinforcing}, $t' \in$ {same or alt, same, alt}, $f' \in$ {reinforcing or non-reinforcing, reinforcing, non-reinforcing}

set of features. Features are generated for all combinations of a, t, t', f and f' for each row. For example, one of the features in the first row is *Percent of neighbors with polarity type positive, that are related via a reinforcing frame relation*. Thus, each feature for the relational classifier identifies neighbors for a given instance via a specific relation (f, t, f' or t', obtained from the scheme annotations) and factors in their polarity values (a, obtained from the classifier predictions from the previous round). This adds a total of 59 relational features to the already existing local features.

ICA has two main phases: the bootstrapping and iterative phases. In the bootstrapping phase, the polarity of each instance is initialized to the most likely value given only the local classifier and its features. In the iterative phase, we create a random ordering of all the instances and, in turn, apply the relational classifier to each instance where the relational features, for a given instance, are computed using the most recent polarity assignments of its neighbors. We repeat this until some stopping criterion is met. For our experiments, we use a fixed number of 30 iterations, which has been found to be sufficient in most data sets for ICA to converge to a solution.

The pseudocode for the algorithm is shown in Algorithm 1.

4.3 Integer Linear Programming

First, we explain the intuition behind viewing discourse relations as enforcing constraints on polarity interpretation. Then, we explain how the constraints are encoded in the optimization problem.

4.3.1 Discourse Constraints on Polarity

The discourse relations between opinions can provide coherence constraints on the way their polarity is interpreted. Consider a discourse scenario in which a speaker expresses multiple opinions regarding the same thing, and is reinforcing his stance in the process (as in Example 1). The set of individual polarity assignments that is most coherent with this global scenario is the one where all the opinions have the same (*equal*) polarity. On the other hand, a pair of individual polarity assignments most consistent with a discourse scenario where a speaker reinforces his stance via opinions towards alternative options, is one with opinions having mutually *opposite* polarity. For instance, in the utterance "Shapes **should** be *curved*, **nothing** *square-like*", the speaker reinforces his pro-curved stance via his opinions about the alternative shapes: *curved* and *square-like*. And, we see that the first opinion is positive and the second is negative. Table 2 lists the discourse relations (target and frame relation combinations) found in the corpus, and the likely polarity interpretation for the related instances.

Target relation + Frame relation	Polarity
same+reinforcing	equal (e)
same+non-reinforcing	opposite (o)
alternative+reinforcing	opposite (o)
alternative+non-reinforcing	equal (e)

Table 2: Discourse relations and their polarity constraints on the related instances.

4.3.2 Optimization Problem

For each DA instance i in a dataset, the local classifier provides a class distribution $[p_i, q_i, r_i]$, where p_i, q_i and r_i correspond to the probabilities that i belongs to positive, negative and neutral categories, respectively. The optimization problem is formulated as an ILP minimization of the objective function in Equation 1.

$$-1 \times \sum_i (p_i x_i + q_i y_i + r_i z_i) + \sum_{i,j} \epsilon_{ij} + \sum_{i,j} \delta_{ij} \quad (1)$$

where the x_i, y_i and z_i are binary *class variables* corresponding to positive, negative and neutral classes, respectively. When a class variable is 1, the corresponding class is chosen. Variables ϵ_{ij} and δ_{ij} are binary *slack variables* that correspond to the discourse constraints between two distinct DA instances i and j. When a given slack variable is 1, the corresponding discourse constraint is violated. Note that the objective function tries to achieve two goals. The first part ($\sum_i p_i x_i + q_i y_i + r_i z_i$) is a maximization that tries to choose a classification for the instances that maximizes the probabilities provided by the local classifier. The second part ($\sum_{i,j} \epsilon_{ij} + \sum_{i,j} \delta_{ij}$) is a minimization that tries to minimize the number of slack variables used, that is, minimize the number of discourse constraints violated.

Constraints in Equations 2 and 3 listed below impose binary constraints on the variables. The constraint in Equation 4 ensures that, for each instance i, only one class variable is set to 1.

$$x_i \in \{0,1\}, y_i \in \{0,1\}, z_i \in \{0,1\}\ ,\ \forall i \quad (2)$$

$$\epsilon_{ij} \in \{0,1\}, \delta_{ij} \in \{0,1\}\ ,\ \forall i \neq j \quad (3)$$

$$x_i + y_i + z_i = 1\ ,\ \forall i \quad (4)$$

We pair distinct DA instances i and j as ij, and if there exists a discourse relation between them, they can be subject to the corresponding polarity constraints listed in Table 2. For this, we define two binary discourse-constraint constants: the *equal-polarity* constant, e_{ij} and the *opposite-polarity* constant, o_{ij}. If a given DA pair ij is related by either a same+reinforcing relation or an alternative+non-reinforcing relation (rows 1, 4 of Table 2), then $e_{ij} = 1$; otherwise it is zero. Similarly, if it is related by either a same+non-reinforcing relation or an alternative+reinforcing relation (rows 2, 3 of Table 2), then $o_{ij} = 1$. Both e_{ij} and o_{ij} are zero if the instance pair is unrelated in the discourse.

For each DA instance pair ij, equal-polarity constraints are applied to the polarity variables of i (x_i, y_i) and j (x_j, y_j) via the following equations:

$$|x_i - x_j| \leq 1 - e_{ij} + \epsilon_{ij}\ ,\ \forall i \neq j \quad (5)$$

$$|y_i - y_j| \leq 1 - e_{ij} + \epsilon_{ij}\ ,\ \forall i \neq j \quad (6)$$

$$-(x_i + y_i) \leq -l_i\ ,\ \forall i \quad (7)$$

When $e_{ij} = 1$, the Equation 5 constrains x_i and x_j to be of the same value (both zero or both one). Similarly, Equation 6 constrains y_i and y_j to be of the same value. Via these equations, we ensure that the instances i and j do not have the opposite polarity when $e_{ij} = 1$. However, notice that, if we use just Equations 5 and 6, the optimization can converge to the same, non-polar (neutral) category. To guide the convergence to the same polar (positive or negative) category, we use Equation 7. Here $l_i = 1$ if the instance i participates in one or more discourse relations. When $e_{ij} = 0$, x_i and x_j (and y_i and y_j), can take on assignments independently of one another. Notice that both constraints 5 and 6 are relaxed when $\epsilon_{ij} = 1$; thus, x_i and x_j (or y_i and y_j) can take on values independently of one another, even if $e_{ij} = 1$.

Next, the opposite-polarity constraints are applied via the following equations:

$$|x_i + x_j - 1| \leq 1 - o_{ij} + \delta_{ij}\ ,\ \forall i \neq j \quad (8)$$

$$|y_i + y_j - 1| \leq 1 - o_{ij} + \delta_{ij}\ ,\ \forall i \neq j \quad (9)$$

In the above equations, when $o_{ij} = 1$, x_i and x_j (and y_i and y_j) take on opposite values; for example, if $x_i = 1$ then $x_j = 0$ and vice versa. When $o_{ij} = 0$, the variable assignments are independent of one another. This set of constraints is relaxed when $\delta_{ij} = 1$.

In general, in our ILP formulation, notice that if an instance does not have a discourse relation to any other instance in the data, its classification is unaffected by the optimization. Also, as the underlying discourse scheme poses constraints only on the interpretation of the polarity of the related instances, discourse constraints are applied only to the polarity variables x and y, and not to the neutral class variable, z. Finally, even though slack variables are used, we discourage the ILP system from indiscriminately setting the slack variables to 1 by making them a part of the objective function that is minimized.

5 Experiments

In this work, we are particularly interested in improvements due to discourse-based methods. Thus, we report performance under three conditions: over only those instances that are related via discourse relations (*Connected*), over instances not related via discourse relations (*Singletons*), and over all instances (*All*).

The annotated data consists of 7 scenario-based, multi-party meetings from the AMI meeting corpus. We filter out very small DAs (DAs with fewer than 3 tokens, punctuation included). This gives

us a total of 4606 DA instances, of which 1935 (42%) have opinion annotations. For our experiments, the DAs with no opinion annotations as well as those with neutral opinions are considered as neutral. Table 3 shows the class distributions in the data for the three conditions.

	Pos	Neg	Neutral	Total
Connected	643	343	81	1067
Singleton	553	233	2753	3539
All	1196	576	2834	4606

Table 3: Class distribution over connected, single and all instances.

5.1 Classifiers

Our first baseline, *Base*, is a simple distribution-based classifier that classifies the test data based on the overall distribution of the classes in the training data. However, in Table 3, the class distribution is different for the Connected and Singleton conditions. We incorporate this in a smarter baseline, *Base-2*, which constructs separate distributions for connected instances and singletons. Thus, given a test instance, depending on whether it is connected, Base-2 uses the corresponding distribution to make its prediction.

The third baseline is the supervised classifier, *Local*, described in Section 4.1. It is implemented using the SVM classifiers from the Weka toolkit (Witten and Frank, 2002).[4] Our supervised discourse-based classifier, ICA from Section 4.2, also uses a similar SVM implementation for its relational classifier. We implement our ILP approach from Section 4.3 using the optimization toolbox from Mathworks (http://www.mathworks.com) and GNU Linear Programming Kit.

We observed that the ILP system performs better than the ICA system on instances that are connected, while ICA performs better on singletons. Thus, we also implemented a simple hybrid classifier (HYB), which selects the ICA prediction for classification of singletons and the ILP prediction for classification of connected instances.

5.2 Results

We performed 7-fold cross validation experiments, where six meetings are used for training and the seventh is used for testing the supervised classifiers (Base, Base-2, Local and ICA). In the case of ILP, the optimization is applied to the output of Local for each test fold. Table 4 reports the accuracies of the classifiers, averaged over 7 folds.

First, we observe that Base performs poorly over connected instances, but performs considerably better over singletons. This is expected as the overall majority class is neutral and the singletons are more likely to be neutral. Base-2, which incorporates the differentiated distributions, performs substantially better than Base. Local achieves an overall performance improvement over Base and Base-2 by 23 percentage points and 9 percentage points, respectively. In general, Local outperforms Base for all three conditions ($p < 0.001$), and Base-2 for the Singleton and All conditions ($p < 0.001$). This overall improvement in Local's accuracy corroborates the utility of the lexical, unigram and DA based features for polarity detection in this corpus.

Turning to the discourse-based classifiers, ICA, ILP and HYB, all of these perform better than Base and Base-2 for all conditions. ICA improves over Local by 9 percentage points for Connected, 3 points for Singleton and 4 points for All. ILP's improvement over Local for Connected and All is even more substantial: 28 percentage points and 6 points, respectively. Notice that ILP has the same performance as Local for Singletons, as the discourse constraints are not applied over unconnected instances. Finally, HYB significantly outperforms Local under all conditions. The significance levels of the improvements over Local are highlighted in Table 4. These improvements also signify that the underlying discourse scheme is effective, and adaptable to different implementations.

Interestingly, ICA and ILP improve over Local in different ways. While ILP sharply improves the performance over the connected instances, ICA shows relatively modest improvements over both connected and singletons. ICA's improvement over singletons is interesting because it indicates that, even though the features in Table 1 are focused on discourse relations, ICA utilizes them to learn the classification of singletons too.

Comparing our discourse-based approaches, ILP does significantly better than ICA over connected instances ($p < 0.001$), while ICA does significantly better than ILP over singletons ($p <$

[4]We use the SMO implementation, which, when used with the logistic regression, has an output that can be viewed as a posterior probability distribution.

	Base	Base-2	Local	ICA	ILP	HYB
Connected	24.4	47.56	46.66	**55.64**	**75.07**	**75.07**
Singleton	51.72	63.23	75.73	78.72	75.73	78.72
All	45.34	59.46	68.72	**73.31**	**75.35**	**77.72**

Table 4: Accuracies of the classifiers measured over Connected, Singleton and All instances. Performance significantly better than Local are indicated in **bold** for $p < 0.001$ and underline for $p < 0.01$.

0.01). However, there is no significant difference between ICA and ILP for the All condition. The HYB classifier outperforms ILP for the Singleton condition ($p < 0.01$) and ICA for the Connected condition ($p < 0.001$). Interestingly, over all instances (the All condition), HYB also performs significantly better than *both* ICA ($p < 0.001$) and ILP ($p < 0.01$).

5.3 Analysis

Amongst our two approaches, ILP performs better, and hence we further analyze its behavior to understand how the improvements are achieved. Table 5 reports the performance of ILP and Local for the precision, recall and f-measure metrics (averaged over 7 test folds), measured separately for each of the opinion categories. The most prominent improvement by ILP is observed for the recall of the polar categories under the Connected condition: 40 percentage points for the positive class, and 29 percentage points for the negative class. The gain in recall is not accompanied by a significant loss in precision. This results in an improvement in f-measure for the polar categories (24 points for positive and 16 points for negative). Also note that, by virtue of the constraint in Equation 7, ILP does not classify any connected instance as neutral; thus the precision is NaN, recall is 0 and the f-meaure is NaN. This is indicated as * in the Table.

The improvement of ILP for the All condition, for the polar classes, follows a similar trend for recall (18 to 21 point improvement) and f-measure (9 to 13 point improvement). In addition to this, the ILP has an *overall improvement in precision over Local*. This may seem counterintuitive, as in Table 5, ILP's precision for connected nodes is similar to, or lower than, that of Local. This is explained by the fact that, while going from connected to overall conditions, Local's polar predictions increase by threefold (565 to 1482), but its *correct* polar predictions increase by only twofold (430 to 801). Thus, the ratio of change in the total polar predictions to the correct polar predictions is $3 : 2$. On the other hand, while polar predictions by ILP increase by only twofold (1067 to 1984), its *correct* polar predictions increase by 1.5 times (804 to 1175). Here, the ratio of change in the total polar predictions to the correct polar predictions is $4 : 3$, a smaller ratio.

Gold	Local			
	Pos	Neg	Neut	Total
Pos	551	113	532	1196
Neg	121	250	205	576
Neut	312	135	2387	2834
Total	984	498	3124	4606
Gold	ILP			
	Pos	Neg	Neut	Total
Pos	817	157	222	1196
Neg	147	358	71	576
Neut	358	147	2329	2834
Total	1322	662	2622	4606

Table 6: Contingency table over all instances.

The contingency table (Table 6) shows how Local and ILP compare against the gold standard annotations. Notice here, that even though ILP makes more polar guesses as compared to Local, a greater proportion of the ILP guesses are correct. The number of non-diagonal elements are much smaller for ILP, resulting in the accuracy improvements seen in Table 4.

6 Examples and Discussion

The results in Table 4 show that Local, which provides the classifications for bootstrapping ICA and ILP, predicts an incorrect class for more than 50% of the connected instances. Methods starting with noisy starting points are in danger of propagating the errors and hence worsening the performance. Interestingly, in spite of starting with so many bad classifications, ILP is able to achieve a large performance improvement. We discovered that, given a set of connected instances, even when Local has only one correct guess, ILP is able to use this to rectify the related instances. We illustrate this situation in Figure 2, which reproduces the connected DAs for Example 1. It shows the classifications

	Positive		Negative		Neutral	
	Local	ILP	Local	ILP	Local	ILP
Connected-Prec	78.1	78.2	71.9	69.8	12.1	
Connected-Recall	45.3	**86.3**	44.1	**73.4**	62.8	*
Connected-F1	56.8	**81.5**	54.0	**70.7**	18.5	
All-Prec	56.2	**61.3**	52.3	*54.6*	76.3	**88.3**
All-Recall	46.6	**67.7**	44.3	**62.5**	83.9	81.5
All-F1	50.4	**64.0**	46.0	**57.1**	79.6	84.6

Table 5: Precision, Recall, Fmeasure for each Polarity category. Performance significantly better than Local are indicated in **bold** ($p < 0.001$), underline ($p < 0.01$) and *italics* ($p < 0.05$). The * denotes that ILP does not retrieve any connected node as neutral.

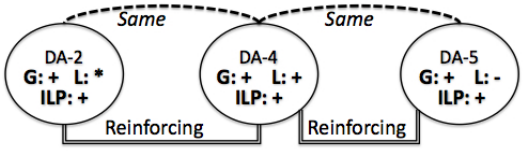


Figure 2: Discourse Relations and Classifications for Example 1.

for each DA from the gold standard (G), the Local classifier (L) and the ILP classifier (ILP). Observe that Local predicts the correct positive class (+) for only DA-4 (the DA containing **bit more durable** and **ergonomic**). Notice that these are clear cases of positive evaluation. It incorrectly predicts the polarity of DA-2 (containing **bit more bouncy**) as neutral (*), and DA-5 (containing **a bit different from all the other remote controls**) as negative (-). DA-2 and DA-5 exemplify the fact that polarity classification is a complex and difficult problem: being bouncy is a positive evaluation in this particular discourse context, and may not be so elsewhere. Thus, naturally, lexicons and unigram-based learning would fail to capture this positive evaluation. Similarly, "being different" could be deemed negative in other discourse contexts. However, ILP is able to arrive at the correct predictions for all the instances. As the DA-4 is connected to both DA-2 and DA-5 via a discourse relation that enforces an equal-polarity constraint (same+reinforcing relation of row 1, Table 2), both of the misclassifications are rectified. Presumably, the incorrect predictions made by Local are low confidence estimates, while the predictions of the correct cases have high confidence, which makes it possible for ILP to make the corrections.

We also observed the propagation of the correct classification for other types of discourse relations, for more complex types of connectivity, and also for conditions where an instance is not directly connected to the correctly predicted instance. The meeting snippet below (Example 2) and its corresponding DA relations (Figure 3) illustrate this. This example is a reinforcing discourse where the speaker is arguing for the number keypad, which is an alternative to the scrolling option. Thus, he argues against the scrolling, and argues for entering the number (which is a capability of the number keypad).

(2) D-1: I reckon you're **gonna have to have** a *number keypad* anyway for the amount of channels these days,
D-2: You **wouldn't want to** just have to *scroll through* all the channels to get to the one you want
D-3: You **wanna** *enter just the number of it* , if you know it
D-4: I reckon **we're gonna have to have** a *number keypad* anyway

In Figure 3, we see that, DA-2 is connected via an alternative+reinforcing discourse relation to each of its neighbors DA-1 and DA-3, which encourages the optimization to choose a class for it that is opposite to DA-1 and DA-3. Notice also, that even though Local predicts only DA-4 correctly, this correct classification finally influences the correct choice for all the instances, including the remotely connected DA-2.

7 Conclusions and Future Work

This work focuses on the first step to ascertain whether discourse relations are useful for improving opinion polarity classification, whether they can be modeled and what modeling choices can be used. To this end, we explored two distinct paradigms: the supervised ICA and the unsupervised ILP. We showed that both of our approaches are effective in exploiting discourse relations to

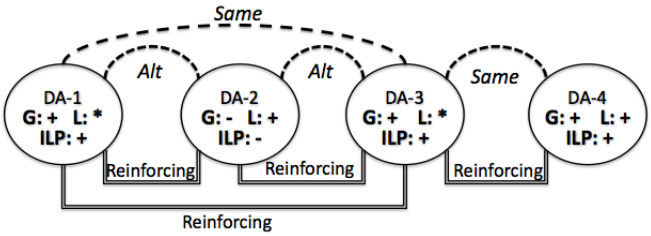


Figure 3: Discourse Relations and Classifications for Example 2.

significantly improve polarity classification. We found that there is a difference in how ICA and ILP achieve improvements, and that combining the two in a hybrid approach can lead to further overall improvement. Quantitatively, we showed that our approach is able to achieve a large increase in recall of the polar categories without harming the precision, which results in the performance improvements. Qualitatively, we illustrated how, even if the bootstrapping process is noisy, the optimization and discourse constraints effectively rectify the misclassifications. The improvements of our diverse global inference approaches indicate that discourse information can be adapted in different ways to augment and improve existing opinion analysis techniques.

The automation of the discourse-relation recognition is the next step in this research. The behavior of ICA and ILP can change, depending on the automation of discourse level recognition. The implementation and comparison of the two methods under full automation is the focus of our future work.

Acknowledgments

This research was supported in part by the Department of Homeland Security under grant N000140710152 and NSF Grant No. 0746930. We would also like to thank the anonymous reviewers for their helpful comments.

References

N. Asher, F. Benamara, and Y. Mathieu. 2008. Distilling opinion in discourse: A preliminary study. *COLING-2008*.

M. Bansal, C. Cardie, and L. Lee. 2008. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. In *COLING-2008*.

M. Bilgic, G. M. Namata, and L. Getoor. 2007. Combining collective classification and link prediction. In *Workshop on Mining Graphs and Complex Structures at the IEEE International Conference on Data Mining*.

J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, I. McCowan, W. Post, D. Reidsma, and P. Wellner. 2005. The ami meetings corpus. In *Proceedings of the Measuring Behavior Symposium on "Annotating and measuring Meeting Behavior"*.

Y. Choi, E. Breck, and C. Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *EMNLP 2006*.

P. Denis and J. Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *HLT-NAACL 2007*.

A. Devitt and K. Ahmad. 2007. Sentiment polarity identification in financial news: A cohesion-based approach. In *ACL 2007*.

A. B. Goldberg and X. Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL 2006 Workshop on Textgraphs: Graph-based Algorithms for Natural Language Processing*.

A. Haghighi, K. Toutanova, and C. Manning. 2005. A joint model for semantic role labeling. In *CoNLL*.

H. Kanayama and T. Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *EMNLP-2006*, pages 355–363, Sydney, Australia.

A. Kennedy and D. Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110–125.

Q. Lu and L. Getoor. 2003. Link-based classification. In *Proceedings of the International Conference on Machine Learning (ICML)*.

A. Moschitti, D. Pighin, and R. Basili. 2006. Semantic role labeling via tree kernel joint inference. In *CoNLL*.

A. Moschitti. 2009. Syntactic and semantic kernels for short text pair categorization. In *EACL*.

D. Neiberg, K. Elenius, and K. Laskowski. 2006. Emotion recognition in spontaneous speech using gmms. In *INTERSPEECH 2006 ICSLP*.

J. Neville and D. Jensen. 2000. Iterative classification in relational data. In *In Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20. AAAI Press.

B. Pang and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACl 2004*.

L. Polanyi and A. Zaenen, 2006. *Contextual Valence Shifters*. Computing Attitude and Affect in Text: Theory and Applications.

A.-M. Popescu and O. Etzioni. 2005. Extracting product features and opinions from reviews. In *HLT-EMNLP 2005*.

S. Raaijmakers, K. Truong, and T. Wilson. 2008. Multimodal subjectivity analysis of multiparty conversation. In *EMNLP*.

M. Richardson and P. Domingos. 2006. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNLL-2004*, pages 1–8. Boston, MA, USA.

K. Sadamitsu, S. Sekine, and M. Yamamoto. 2008. Sentiment analysis based on probabilistic models using inter-sentence information. In *LREC'08*.

S. Somasundaran, J. Ruppenhofer, and J. Wiebe. 2007. Detecting arguing and sentiment in meetings. In *SIGdial Workshop on Discourse and Dialogue 2007*.

S. Somasundaran, J. Wiebe, and J. Ruppenhofer. 2008. Discourse level opinion interpretation. In *Coling 2008*.

H. Takamura, T. Inui, and M. Okumura. 2007. Extracting semantic orientations of phrases from dictionary. In *HLT-NAACL 2007*.

B. Taskar, M. Wong, P. Abbeel, and D. Koller. 2004. Link prediction in relational data. In *Neural Information Processing Systems*.

M. Thomas, B. Pang, and L. Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *EMNLP 2006*.

T. Wilson, J. Wiebe, and P. Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT-EMNLP 2005*.

I. H. Witten and E. Frank. 2002. Data mining: practical machine learning tools and techniques with java implementations. *SIGMOD Rec.*, 31(1):76–77.

Sentiment Analysis of Conditional Sentences

Ramanathan Narayanan
Dept. of EECS
Northwestern University
ramanathan.an@gmail.com

Bing Liu *
Dept. of Computer Science
Univ. of Illinois at Chicago
liub@cs.uic.edu

Alok Choudhary
Dept. of EECS
Northwestern University
alokchoudhary01@gmail.com

Abstract

This paper studies sentiment analysis of conditional sentences. The aim is to determine whether opinions expressed on different topics in a conditional sentence are positive, negative or neutral. Conditional sentences are one of the commonly used language constructs in text. In a typical document, there are around 8% of such sentences. Due to the condition clause, sentiments expressed in a conditional sentence can be hard to determine. For example, in the sentence, *if your Nokia phone is not good, buy this great Samsung phone*, the author is positive about "*Samsung phone"* but does not express an opinion on "*Nokia phone"* (although the owner of the "*Nokia phone*" may be negative about it). However, if the sentence does not have "*if*", the first clause is clearly negative. Although "*if*" commonly signifies a conditional sentence, there are many other words and constructs that can express conditions. This paper first presents a linguistic analysis of such sentences, and then builds some supervised learning models to determine if sentiments expressed on different topics in a conditional sentence are positive, negative or neutral. Experimental results on conditional sentences from 5 diverse domains are given to demonstrate the effectiveness of the proposed approach.

1 Introduction

Sentiment analysis (also called opinion mining) has been an active research area in recent years. There are many research directions, e.g., sentiment classification (classifying an opinion document as positive or negative) (e.g., Pang, Lee and Vaithyanathan, 2002; Turney, 2002), subjectivity classification (determining whether a sentence is subjective or objective, and its associated opinion) (Wiebe and Wilson, 2002; Yu and Hatzivassiloglou, 2003; Wilson et al, 2004; Kim and Hovy, 2004; Riloff and Wiebe, 2005), feature/topic-based sentiment analysis (assigning positive or negative sentiments to topics or product features) (Hu and Liu 2004; Popescu and Etzioni, 2005; Carenini et al., 2005; Ku et al., 2006; Kobayashi, Inui and Matsumoto, 2007; Titov and McDonald. 2008). Formal definitions of different aspects of the sentiment analysis problem and discussions of major research directions and algorithms can be found in (Liu, 2006; Liu, 2009). A comprehensive survey of the field can be found in (Pang and Lee, 2008).

Our work is in the area of topic/feature-based sentiment analysis or opinion mining (Hu and Liu, 2004). The existing research focuses on solving the general problem. However, we argue that it is unlikely to have a one-technique-fit-all solution because different types of sentences express sentiments/opinions in different ways. A divide-and-conquer approach is needed, e.g., focused studies on different types of sentences. This paper focuses on one type of sentences, i.e., conditional sentences, which have some unique characteristics that make it hard to determine the orientation of sentiments on topics/features in such sentences. By *sentiment orientation*, we mean positive, negative or neutral opinions. By *topic*, we mean the target on which an opinion has been expressed. In the product domain, a topic is usually a *product feature* (i.e., a component or attribute). For example, in the sentence, *I do not like the sound quality, but love the design of this MP3 player*, the product features (topics) are "*sound quality*" and "*design*" of the MP3 player as opinions have been expressed on them. The sentiment is positive on "*design*" but negative on "*sound quality*".

Conditional sentences are sentences that describe implications or hypothetical situations and their consequences. In the English language, a variety of conditional connectives can be used to form these sentences. A conditional sentence contains two clauses: the condition clause and

* This work was done when Bing Liu was on sabbatical leave at Northwestern University.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 180–189,
Singapore, 6-7 August 2009.

the consequent clause, that are dependent on each other. Their relationship has significant implications on whether the sentence describes an opinion. One simple observation is that sentiment words (also known as opinion words) (e.g., *great*, *beautiful*, *bad*) alone cannot distinguish an opinion sentence from a non-opinion one. A conditional sentence may contain many sentiment words or phrases, but express no opinion.

Example 1: *If someone makes a beautiful and reliable car, I will buy it* expresses no sentiment towards any particular car, although "*beautiful*" and "*reliable*" are positive sentiment words.

This, however, does not mean that a conditional sentence cannot express opinions/sentiments.

Example 2: *If your Nokia phone is not good, buy this great Samsung phone* is positive about the "*Samsung phone*" but does not express an opinion on the "*Nokia phone*" (although the owner of the "*Nokia phone*" may be negative about it). Clearly, if the sentence does not have "*if*", the first clause is negative. Hence, a method for determining sentiments in normal sentences will not work for conditional sentences. The examples below further illustrate the point.

In many cases, both the condition and consequent together determine the opinion.

Example 3: *If you are looking for a phone with good voice quality, don't buy this Nokia phone* is negative about the "*voice quality*" of the "Nokia phone", although there is a positive sentiment word "*good*" in the conditional clause modifying "*voice quality*". However, in the following example, the opinion is just the opposite.

Example 4: *If you want a phone with good voice quality, buy this Nokia phone* is positive about the "*voice quality*" of the "Nokia phone".

As we can see, sentiment analysis of conditional sentences is a challenging problem.

One may ask whether there is a large percentage of conditional sentences to warrant a focused study. Indeed, there is a fairly large proportion of such sentences in evaluative text. They can have a major impact on the sentiment analysis accuracy. Table 1 shows the percentage of conditional sentences (sentences containing the words *if*, *unless*, *assuming*, etc) and also the total number of sentences from which we computed the percentage in several user-forums. The figures definitely suggest that there is considerable benefit to be gained by developing techniques that can analyze conditional sentences.

Table 1: Percent of conditional sentences

Source	% of cond. (total #. of sent.)
Cellphone	8.6 (47711)
Automobile	5.0 (8113)
LCD TV	9.92 (258078)
Audio Systems	8.1 (5702)
Medicine	8.29 (160259)

To the best of our knowledge, there is no focused study on conditional sentences. This paper makes such an attempt. Specifically, we determine whether a conditional sentence (which is also called a *conditional* in the linguistic literature) expresses positive, negative or neutral opinions on some topics/features. Since our focus is on studying how conditions and consequents affect sentiments, we assume that *topics* are given, which are *product attributes* since our data sets are user comments on different products.

Our study is conducted from two perspectives. We start with the linguistic angle to gain a good understanding of existing work on different types of conditionals. As conditionals can be expressed with other words or phrases than *if*, we will study how they behave compared to *if*. We will also show that the distribution of these conditionals based on our data sets.

With the linguistic knowledge, we perform a computational study using machine learning. A set of features for learning is designed to capture the essential determining information. Note that the features here are data attributes used in learning rather than product attributes or features. Three classification strategies are designed to study how to best perform the classification task due to the complex situation of two clauses and their interactions in conditional sentences. These three classification strategies are *clause-based*, *consequent-based* and *whole-sentence-based*. Clause-based classification classifies each clause separately and then combines their results. Consequent-based classification only uses consequents for classification as it is observed that in conditional sentences, it is often the consequents that decide the opinion. Whole-sentence-based classification treats the entire sentence as a whole in classification. Experimental results on conditional sentences from diverse domains demonstrate the effectiveness of these classification models. The results indicate that the whole-sentence-based classifier performs the best.

Since this paper only studies conditional sentences, a natural question is whether the proposed technique can be easily integrated into an overall sentiment analysis or opinion mining system. The answer is yes because a large proportion of conditional sentences can be detected using conditional connectives. Keyword search is

thus sufficient to identify such sentences for special handling using the proposed approach. There are, however, some subtle conditionals which do not use normal conditional connectives and will need an additional module to identify them, but such sentences are very rare as Table 2 indicates.

2 The Problem Statement

The paper follows the *feature-based sentiment analysis* model in (Hu and Liu 2004; Popescu and Etzioni, 2005). We are particularly interested in sentiments on products and services, which are called objects or entities. Each object is described by its *parts* and *attributes*, which are collectively called *features* in (Hu and Liu, 2004; Liu, 2006). For example, in the sentence, *If this camera has great picture quality, I will buy it*, "*picture quality*" is a feature of the camera. For formal definitions of objects and features, please refer to (Liu, 2006; Liu, 2009). In this paper, we use the term *topic* to mean *feature* as the feature here can confuse with the feature used in machine learning. The term topic has also been used by some researchers (e.g., Kim and Hovy, 2004; Stoyanov and Cardie, 2008).

Our objective is to predict the sentiment orientation (positive, negative or neutral) on each topic that has been commented on in a sentence.

The problem of automatically identifying features or topics being spoken about in a sentence has been studied in (Hu and Liu, 2004; Popescu and Etzioni, 2005; Stoyanov and Cardie, 2008). In this work, we do not attempt to identify such topics automatically. Instead, we assume that they are given because our objective is to study how the interaction of the condition and consequent clauses affects sentiments. For this purpose, we manually identify all the topics.

3 Conditional Sentences

This section presents the linguistic perspective of conditional sentences.

Table 2: Percentage of sentences with some main conditional connectives

Conditional Connective	% of sentences
If	6.42
Unless	0.32
Even if	0.17
Until	0.10
As (so) long as	0.09
Assuming/supposing	0.04
In case	0.04
Only if	0.03

3.1 Conditional Connectives

A large majority of conditional sentences are introduced by the subordinating conjunction *If*. However, there are also many other conditional connectives, e.g., *even if*, *unless*, *in case*, *assuming*/*supposing*, *as long as*, etc. Table 2 shows the distribution of conditional sentences with various connectives in our data. Detailed linguistic discussions of them are beyond the scope of this paper. Interested readers, please refer to (Declerck and Reed, 2001). Below, we briefly discuss some important ones and their interpretations.

If: This is the most commonly used conditional connective. In addition to its own usage, it can also be used to replace other conditional connectives, except some semantically richer connectives (Declerck and Reed, 2001). Most (but not all) conditional sentences can be logically expressed in the form 'If *P* then *Q*', where *P* is the condition clause and *Q* is the consequent clause. For practical purposes, we can automatically segment the condition and consequent clauses using simple rules generated by observing grammatical and linguistic patterns.

Unless: Most conditional sentences containing *unless* can be replaced with equivalent sentences with an *if* and a *not*. For example, the sentence *Unless you need clarity, buy the cheaper model* can be expressed with *If you don't need clarity, buy the cheaper model*.

Even if: Linguistic theories claim that *even if* is a special case of a conditional which may not always imply an if-then relationship (Gauker 2005). However, in our datasets, we have observed that the usage of *even if* almost always translates into a conditional. Replacing *even if* by *if* will yield a sentence that is semantically similar enough for the purpose of sentiment analysis.

Only if, provided/providing that, on condition that: Conditionals involving these phrases typically express a necessary condition, e.g., *I will buy this camera only if they can reduce the price*. In such sentences, *only* usually does not affect whether the sentence is opinionated or not.

In case: Conditional sentences containing *in case* usually describe a precaution (*I will close the window in case it rains*), prevention (*I wore sunglasses in case I was recognized*), or a relevance conditional (*In case you need a car, you can rent one*). Identifying the conditional and consequent clauses is not straightforward in many cases. Further, in these instances, replacing *in case* with *if* may not convey the intended meaning of the conditional. We have ignored these cases in

our analysis as we believe that they need a separate study, and also such sentences are rare.

As (so) long as: Sentences with these connectives behave similarly to *if* and can usually be replaced with *if*.

Assuming/Supposing: These are a category of conditionals that behave quite differently. The participles *supposing* and *assuming* create conditional sentences where the conditional clause and the consequent clause can be syntactically independent. It is quite difficult to distinguish those conditional sentences which contain an explicit consequent clause and fit within our analysis framework. In our data, most of such sentences have no consequent, thus representing assumptions rather than opinions. We omit these sentences in our study (they are also rare).

3.2 Types of Conditionals

There are extensive studies of conditional sentences (also known as *conditionals*) in linguistics. Various theories have led to a number of classification systems. Popular types of conditionals include *actualization conditionals, inferential conditionals*, *implicative conditionals*, etc (Declerck and Reed, 2001). However, these classifications are mainly based on semantic meanings which are difficult to recognize by a computer program. To build classification models, we instead exploit *canonical tense patterns* of conditionals, which are often used in pedagogic grammar books. They are defined based on tense and are associated with general meanings. However, as described in (Declerck and Reed, 2001), their meanings are much more complex and numerous than their associated general meanings. However, the advantage of this classification is that different types can be detected easily because they depend on tense which can be produced by a part-of-speech tagger. As we will see in Section 5, canonical tense patterns help sentiment classification significantly. Below, we introduce the four canonical tense patterns.

Zero Conditional: This conditional form is used to describe universal statements like facts, rules and certainties. In a zero conditional, both the condition and consequent clauses are in the simple present tense. An example of such sentences is: *If you heat water, it boils*.

First Conditional: Conditional sentences of this type are also called potential or indicative conditionals. They are used to express a hypothetical situation that is probably true, but the truth of which is unverified. In the first conditional, the condition is in the simple present tense, and the consequent can be either in past tense or present tense, usually with a modal auxiliary verb preceding the main verb, e.g., *If the acceleration is good, I will buy it.*

Second Conditional: This is usually used to describe less probable situations, for stating preferences and imaginary events. The condition clause of a second conditional sentence is in the past subjunctive (past tense), and the consequent clause contains a conditional verb modifier (like *would*, *should*, *might*), in addition to the main verb, e.g., *If the cell phone was robust, I would consider buying it.*

Third conditional: This is usually used to describe contrary-to-fact (impossible) past events. The past perfect tense is used in the condition clause, and the consequent clause is in the present perfect tense, e.g., *If I had bought the a767, I would have hated it.*

Based on the above definitions, we have developed approximate part-of-speech (POS) tags [1] for the condition and the consequent of each pattern (Table 3), which do not cover all sentences, but overall they cover a majority of the sentences. For those not covered cases, the problem is mainly due to incomplete sentences and wrong grammars, which are typical for informal writings in forum postings and blogs. For example, the sentence, *Great car if you need powerful acceleration*, does not fall into any category, but it actually means *It is a great car if you need powerful acceleration*, which is a zero conditional. To handle such sentences, we designed a set of rules to assign them some default types:

If condition contains VB/VBP/VBZ → 0 conditional
If consequent contains VB/VBP/VBS → 0 conditional
If condition contains VBG → 1st conditional
If condition contains VBD → 2nd conditional
If conditional contains VBN → 3rd conditional.

Table 3: Tenses for identifying conditional types

Type	Linguistic Rule	Condition POS tags	Consequent POS tags
0	If + simple present → simple present	VB/VBP/VBZ	VB/VBP/ VBZ
1	If + simple present → will + bare infinitive	VB/VBP/VBZ /VBG	MD + VB
2	If + past tense → would + infinitive	VBD	MD + VB
3	If + past perfect → present perfect	VBD+VBN	MD + VBD

[1] The list of Part-Of-Speech (POS) tags can be found at: http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

By using these rules, we can increase the sentence coverage from 73% to 95%.

4 Sentiment Analysis of Conditionals

We now describe our computational study. We take a machine learning approach to predict sentiment orientations. Below, we first describe features used and then classification strategies.

4.1 Feature construction

I. *Sentiment words/phrases and their locations*: Sentiment words are words used to express positive or negative opinions, which are instrumental for sentiment classification for obvious reasons. We obtained a list of over 6500 sentiment words gathered from various sources. The bulk of it is from http://www.cs.pitt.edu/mpqa. We also added some of our own. Our list is mainly from the work in (Hu and Liu, 2004; Ding, Liu and Yu, 2008). In addition to words, there are phrases that describe opinions. We have identified a set of such phrases. Although obtaining these phrases was time-consuming, it was only a one-time effort. We will make this list available as a community resource. It is possible that there is a better automated method for finding such phrases, such as the methods in (Kanayama and Nasukawa, 2006; Breck, Choi and Cardie, 2007). However, automatically generating sentiment phrases has not been the focus of this work as our objective is to study how the two clauses interact to determine opinions given the sentiment words and phrases are known. Our list of phrases is by no means complete and we will continue to expand it in the future.

   For each sentence, we also identify whether it contains sentiment words/phrases in its condition or consequent clause. It was observed that the presence of a sentiment word/phrase in the consequent clause has more effect on the sentiment of a sentence.

II. *POS tags of sentiment words*: Sentiment words may be used in several contexts, not all of which may correspond to an opinion. For example, *I trust Motorola* and *He has a trust fund* both contain the word *trust*. But only the former contains an opinion. In such cases, the POS tags can provide useful information.

III. *Words indicating no opinion*: Similar to how sentiment words are related to opinions, there are also a number of words which imply the opposite. Words like *wondering*, *thinking*, *debating* are used when the user is posing a question or expressing doubts. Thus such phrases usually do not contribute an opinion, especially if they are in the vicinity of the *if* connective. We search a window of 3 words on either side of *if* to determine if there is any such word. We have compiled a list of these words as well and use it in our experiments.

IV. *Tense patterns*: These are the canonical tense patterns in Section 3.2. They are used to generate a set of features. We identify the first verb in both the condition and consequent clauses by searching for the relevant POS tags in Table 3. We also search for the words preceding the main verb to find modal auxiliary verbs, which are also used as features.

V. *Special characters*: The presence or absence of '?' and '!'.

VI. *Conditional connectives*: The conditional connective used in the sentence (*if*, *even if*, *unless*, *only if*, etc) is also taken as a feature.

VII. *Length of condition and consequent clauses*: Using simple linguistic and punctuation rules, we automatically segment a sentence into condition and consequent clauses. The numbers of words in the condition and consequent clauses are then used as features. We observed that when the condition clause is short, it usually has no impact on whether the sentence expresses an opinion.

VIII. *Negation words*: The use of negation words like *not, don't*, *never*, etc, often alter the sentiment orientation of a sentence. For example, the addition of *not* before a sentiment word can change the orientation of a sentence from positive to negative. We consider a window of 3-6 words before an opinion word, and search for these kinds of words.

The following two features are singled out for easy reference later. They are only used in one classification strategy. The first feature is an indicator, and the second feature has a parameter (which will be evaluated separately).

(1). *Topic location*: This feature indicates whether the topic is in the conditional clause or the consequent clause.

(2). *Opinion weight*: This feature considers only sentiment words in the vicinity of the topic, since they are more likely to influence the opinion on the topic. A window size is used to control what we mean by vicinity. The following formula is used to assign a weight to each sentiment word, which is inversely proportional to the distance (D_{op}) of the sentiment word to the topic mention. Sentiment

value is +1 for a positive word and -1 for a negative word. *Sentwords* are the set of known sentiment words and phrases.

$$weight = \sum_{op} \frac{\pm 1}{D_{op}}, \forall op \in \{sentwords\}$$

4.2 Classification Strategies

Since we are interested in topic-based sentiment analysis, how to perform classification becomes an interesting issue. Due to the two clauses, it may not be sufficient to classify the whole sentence as positive or negative as in the same sentence, some topics may be positive and some may be negative. We propose three strategies.

Clause-based classification: Since there are two clauses in a conditional sentence, in this case we build two classifiers, one for the condition and one for the consequent.

Condition classifier: This method classifies the condition clause as expressing positive, negative or neutral opinion.

Training data: Each training sentence is represented as a feature vector. Its class is positive, negative or neutral depending on whether the conditional clause is positive, negative or neutral while considering both clauses.

Testing: For each test sentence, the resulting classifier predicts the opinion of the condition clause.

Topic class prediction: To predict the opinion on a topic, if the topic is in the condition clause, it takes the predicted class of the clause.

Consequent classifier: This classifier classifies the consequent clause as expressing positive, negative or neutral opinion.

Training data: Each training sentence is represented as a feature vector. Its class is positive, negative or neutral depending on whether the consequent clause is positive, negative or neutral while considering both clauses.

Testing: For each test sentence, the resulting classifier predicts the opinion of the consequent clause.

Topic class prediction: To predict the opinion on a topic, if the topic is in the consequent clause, it takes the predicted class of the clause.

The combination of these two classifiers is called the *clause-based classifier*. It works as follows: If a topic is in the conditional clause, the condition classifier is used, and if a topic is in the consequent clause, the consequent classifier is used.

Consequent-based classification: It is observed that in most cases, the condition clause contains no opinion whereas the consequent clause reflects the sentiment of the entire sentence. Thus, this method uses (in a different way) only the above consequent classifier. If it classifies the consequent of a testing conditional sentence as positive, all the topics in the whole sentence are assigned the positive orientation, and likewise for negative and neutral.

Whole-sentence-based classification: In this case, a single classifier is built to predict the opinion on each topic in a sentence.

Training data: In addition to the normal features, the two features (1) and (2) in Section 4.1 are used for this classifier. If a sentence contains multiple topics, multiple training instances of the same sentence are created in the training data. Each instance represents one specific topic. The class of the instance depends on whether the opinion on the topic is positive, negative or neutral.

Testing: For each topic in each test sentence, the resulting classifier predicts its opinion.

Topic class prediction: This is not needed as the prediction has been done in testing.

5 Results and Discussions

5.1 Data sets

Our data consists of conditional sentences from 5 different user forums: Cellphone, Automobile, LCD TV, Audio systems and Medicine. We obtained user postings from these forums and extracted the conditional sentences. We then manually annotated 1378 sentences from this corpus. We also annotated the conditional and consequent clauses and identified the topics (or product features) being commented upon, and their sentiment orientations. In our annotation, we observed that sentences with no sentiment words or phrases almost never express opinions, i.e., only around 3% of them express opinions. There are around 26% sentences containing no sentiment words or phrases in our data. To make the problem challenging, we restrict our attention to only those sentences that contain at least one sentiment word or phrase. We have annotated topics from around 900 such sentences. Table 4 shows the class distributions of this data. At the clause level (topics are not considered), we observe that conditional clauses contain few opinions. At the topic-level, 43.5% of the topics have positive opinions, 26.4% of the topics have negative opinions, and the rest have no opinions.

Table 4: Distribution of classes

	Positive	Negative	Neutral
Condition	6.9%	6.7%	86.4%
Consequent	49.3%	16.5%	34%
Topic-level	43.5%	26.4%	29.9%

For the annotation of data, we assume that topics are known. One student annotated the topics first. Then two students annotated the sentiments on the topics. If a student found that a topic annotation is wrong, he will let us know. Some mistakes and missing topics were found but there were mainly due to oversights rather than disagreements. The agreement on sentiment annotations were computed using the Kappa score. We achieved the Kappa score of 0.63, which indicates strong agreements. The conflicting cases were then solved through discussion to reach consensus. We did not find anything that the annotators absolutely disagree with each other.

5.2 Experimental results

We now present the results for different combinations of features and classification strategies. For model building, we used Support Vector Machines (SVM), and the LIBSVM implementation (Chang and Lin, 2001) with a Gaussian kernel, which produces the best results. All the results are obtained via 10-fold cross validation.

Two-class classification: We first discuss the results for a simpler version of the problem that involves only sentences with positive or negative orientations on some topics (at least one of the clauses must have a positive/negative opinion on a topic). Neutral sentences are not used (~28% of the total). The results of all three classifiers are given in Table 5. The feature sets have been described in Section 4.1. For all the experiments below, features (1) and (2) are only used by the whole-sentence-based classifier, but not used by the other two classifiers for obvious reasons.

{I+II}: This setting uses sentiment words and phrases, their positions and POS tags as features (we used Brill's POS tagger). This can be seen as the baseline. We observe that both the consequent-based and whole-sentence-based classifiers perform dramatically better than the clause-based classifier. The consequent-based classifier and the whole-sentence-based classifier perform similarly (with the latter being slightly better). The precision, recall, and F-score are computed as the average of the two classes.

{I+II+III}: In this setting, the list of special non-sentiment related words is added to the feature set. All three classifiers improve slightly.

{I+II+III+IV}: This setting includes all the canonical tense based features. We see marked improvements for the consequent-based and whole-sentence-based classifiers both in term of accuracy and F-score, which are statistically significant compared to those of {I+II+III} at the 95% confidence level based on paired t-test.

All: When all the features are used, the results of all the classifiers improve further.

Two main observations worth mentioning:

1. Both the consequent-based and whole-sentence-based classifiers outperform the clause-based classifier dramatically. This confirms our observation that the consequent usually plays the key role in determining the sentiment of the sentence. This is further reinforced by the fact that the consequent-based classifier actually performs similarly to the whole-sentence-based classifier. The condition clause seems to give no help.
2. The second observation is that the linguistic knowledge of canonical tense patterns helps significantly. This shows that the linguistic knowledge is very useful.

We also noticed that many misclassifications are caused by grammatical errors, use of slang phrases and improper punctuations, which are typical of postings on the Web. Due to language irregularities (e.g., wrong grammar, missing punctuations, sarcasm, exclamations), the POS tagger makes many mistakes as well causing some errors in the tense based features.

Three-class classification: We now move to the more difficult and realistic case of three classes: positive, negative and neutral (no-opinion). Table 6 shows the results. The trend is similar except that the whole-sentence-based classifier now performs markedly better than the consequent-based classifier. We believe that this is because the neutral class needs information from both the condition and consequent clauses. This is evident from the fact that there is little or no improvement after {I+II} for the consequent-based classifier. We also observe that the accuracies and F-scores for the three-class classification are lower than those for the two-class classification. This is understandable due to the difficulty of determining whether a sentence has opinion or not. Again, statistical test shows that the canonical tense-based features help significantly.

As mentioned in Section 4.1, the whole-sentence-based classifier only considers those sentiment words in the vicinity of the topic under

Table 5: Two-class classification – positive and negative

	Clause-based classifier				***Consequent-based classifier***				***Whole-sentence-based classifier***			
	Acc.	Prec.	Rec.	**F**	Acc.	Prec.	Rec.	**F**	Acc.	Prec.	Rec.	**F**
I+II (senti. words+POS)	39.9	42.8	34.0	**37.9**	69.1	72.9	67.1	**69.8**	68.9	73.7	68.13	**70.8**
I+II+III (+ non-senti. words)	41.5	44.9	37.1	**40.6**	69.3	73.9	66.3	**69.9**	69.2	73.7	63.5	**71.0**
I+II+III+IV (+ tenses)	42.7	45.2	38.5	**41.6**	72.7	76.4	72.0	**74.1**	71.1	77.9	72.2	**74.9**
All	43.2	46.1	38.9	**42.2**	73.3	77.0	72.7	**74.8**	72.3	77.8	73.6	**75.6**

Table 6: Three-class classification – positive, negative and neutral (no opinion)

	Clause-based classifier				***Consequent-based classifier***				***Whole-sentence-based classifier***			
	Acc.	Prec.	Rec.	**F**	Acc.	Prec.	Rec.	**F**	Acc.	Prec.	Rec.	**F**
I+II (senti. words+POS)	45.2	41.3	35.1	**37.9**	54.6	57.7	52.9	**55.2**	59.1	58.1	56.4	**57.2**
I+II+III (+ non-senti. words)	46.9	42.8	37.8	**40.1**	55.3	60.0	51.3	**55.3**	61.4	60.1	60.8	**60.4**
I+II+III+IV (+ tenses)	50.3	48.7	40.9	**44.5**	57.3	64.0	50.0	**56.1**	64.6	63.3	63.9	**63.6**
All	53.3	49.8	44.1	**46.8**	58.7	64.5	50.1	**56.4**	67.8	66.9	65.1	**66.0**

Table 7: Accuracy of the whole-sentence-based classifier with varying window sizes (*n*)

Window size	1	2	3	4	5	6	7	**8**	9	10
Accuracy	66.1	62.6	64.1	64.8	65.3	65.7	66.3	**67.3**	66.9	66.8

investigation. For this, we search a window of *n* words on either side of the topic mention. To study the effect of varying *n*, we performed an experiment with various values of the window size and measured the overall accuracy for each case. Table 7 shows how the accuracy changes as we increase the window size. We found that a window size of 6-10 yielded good accuracies. This is because lower values of *n* lead to loss of information regarding sentiment words as some sentiment words could be far from the topic. We finally used 8, which gave the best results.

We also investigated ways of using the negation word in the sentence to correctly predict the sentiment. One method is to use the negation word as a feature, as described in Section 4.1. Another technique is to reverse the orientation of the prediction for those sentences which contain negation words. We found that the former technique yielded better results. The results reported so far are based on the former approach.

6 Related Work

There are several research directions in sentiment analysis (or opinion mining). One of the main directions is sentiment classification, which classifies the whole opinion document (e.g., a product review) as positive or negative (e.g., Pang et al, 2002; Turney, 2002; Dave et al, 2003; Ng et al. 2006; McDonald et al, 2007). It is clearly different from our work as we are interested in conditional sentences.

Another important direction is classifying sentences as subjective or objective, and classifying subjective sentences or clauses as positive or negative (Wiebe et al, 1999; Wiebe and Wilson, 2002, Yu and Hatzivassiloglou, 2003; Wilson et al, 2004; Kim and Hovy, 2004; Riloff and Wiebe, 2005; Gamon et al 2005; McDonald et al, 2007). Although these works deal with sentences, they aim to solve the general problem. This paper argues that there is unlikely a one-technique-fit-all solution, and advocates dealing with specific types of sentences differently by exploiting their unique characteristics. Conditional sentences are the focus of this paper. To the best of our knowledge, there is no focused study on them.

Several researchers also studied feature/topic-based sentiment analysis (e.g., Hu and Liu, 2004; Popescu and Etzioni, 2005; Ku et al, 2006; Carenini et al, 2006; Mei et al, 2007; Ding, Liu and Yu, 2008; Titov and R. McDonald, 2008; Stoyanov and Cardie, 2008; Lu and Zhai, 2008). Their objective is to extract topics or product features in sentences and determine whether the sentiments expressed on them are positive or negative. Again, no focused study has been made to handle conditional sentences. Effectively handling of conditional sentences can help their effort significantly.

In this work, we used many sentiment words and phrases. These words and phrases are usually compiled using different approaches (Hatzivassiloglou and McKeown, 1997; Kaji and Kitsuregawa, 2006; Kanayama and Nasukawa, 2006; Esuli and Sebastiani, 2006; Breck et al, 2007; Ding, Liu and Yu. 2008; Qiu et al, 2009). There are several existing lists produced by researchers. We used the one from the MPQA corpus (http://www.cs.pitt.edu/mpqa) with added phrases of our own from (Ding, Liu and Yu. 2008). In our work, we also assume that the topics are known. (Hu and Liu, 2004; Popescu and Etzioni, 2005; Kobayashi, Inui and Matsumoto, 2007; Stoyanov and Cardie, 2008) have studied topic/feature extraction.

One existing focused study is on comparative and superlative sentences (Jindal and Liu, 2006; Bos and Nissim, 2006; Fiszman et al, 2007; Ganapathibhotla and Liu, 2008). Their work identifies comparative sentences, extracts comparative relations in the sentences and analyzes comparative opinions (Ganapathibhotla and Liu, 2008). An example comparative sentence is "*Honda looks better than Toyota*". As we can see, comparative sentences are entirely different from conditional sentences. Thus, their methods cannot be directly applied to conditional sentences.

7 Conclusion

To perform sentiment analysis accurately, we argue that a divide-and-conquer approach is needed, i.e., focused study on each type of sentences. It is unlikely that there is a one-size-fit-all solution. This paper studied one type, i.e., conditional sentences, which have some unique characteristics that need special handling. Our study was carried out from both the linguistic and computational perspectives. In the linguistic study, we focused on canonical tense patterns, which have been showed useful in classification. In the computational study, we built SVM models to automatically predict whether opinions on topics are positive, negative or neutral. Experimental results have shown the effectiveness of the models.

In our future work, we will further improve the classification accuracy and study related problems, e.g., identifying topics/features. Although there are some special conditional sentences that do not use easily recognizable conditional connectives and identifying them are useful, such sentences are very rare and spending time and effort on them may not be cost-effective at the moment.

Acknowledgements

This work was supported in part by DOE SCI-DAC-2: Scientific Data Management Center for Enabling Technologies (CET) grant DE-FC02-07ER25808, DOE FASTOS award number DE-FG02-08ER25848, NSF HECURA CCF-0621443, NSF SDCI OCI-0724599, and NSF ST-HEC CCF-0444405.

References

J. Bos, and M. Nissim. 2006. An Empirical Approach to the Interpretation of Superlatives. EMNLP-2006.

E. Breck, Y. Choi, and C. Cardie. 2007. Identifying expressions of opinion in context, IJCAI-2007.

C.-C. Chang and C.-J. Lin. 2001. LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw /~cjlin/libsvm

G. Carenini, R. Ng, and A. Pauls. 2006. Interactive Multimedia Summaries of Evaluative Text. IUI-2006.

C. Gauker. 2005. Conditionals in Context. MIT Press.

D. Dave, A. Lawrence, and D. Pennock. 2003. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. WWW-2003.

R. Declerck, and S. Reed. 2001. Conditionals: A Comprehensive Empirical Analysis. Berlin: Mouton de Gruyter.

X. Ding, B. Liu, and P. S. Yu. 2008. A holistic lexicon-based approach to opinion mining. WSDM-2008.

A. Esuli, and F. 2006. Sebastiani. Determining term subjectivity and term orientation for opinion mining, EACL-2006.

M. Fiszman, D. Demner-Fushman, F. Lang, P. Goetz, and T. Rindflesch. 2007. Interpreting Comparative Constructions in Biomedical Text. BioNLP-2007.

M. Gamon, A. Aue, S. Corston-Oliver, S. and E. Ringger. 2005. Pulse: Mining customer opinions from free text. IDA-2005.

G. Ganapathibhotla and B. Liu. 2008. Identifying Preferred Entities in Comparative Sentences. COLING-2008.

V. Hatzivassiloglou, and K. McKeown, K. 1997.

Predicting the Semantic Orientation of Adjectives. ACL-EACL-1997.

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. KDD-2004.

N. Jindal, and B. Liu. 2006. Mining Comparative Sentences and Relations. AAAI-2006.

N. Kaji, and M. Kitsuregawa. 2006. Automatic construction of polarity-tagged corpus from HTML documents. ACL-2006.

H. Kanayama, and T. Nasukawa. 2006. Fully Automatic Lexicon Expansion for Domain-Oriented Sentiment Analysis. EMNLP-2006.

S. Kim and E. Hovy. 2004. Determining the Sentiment of Opinions. COLING-2004.

N. Kobayashi, K. Inui and Y. Matsumoto. 2007. Extracting Aspect-Evaluation and Aspect-of Relations in Opinion Mining**.** EMNLP-2007.

L.-W. Ku, Y.-T. Liang, and H.-H. Chen. 2006, Opinion Extraction, Summarization and Tracking in News and Blog Corpora. AAAI-CAAW.

B. Liu. 2006. Web Data Mining: Exploring Hyperlinks, Content and Usage Data. Springer.

B. Liu. 2009. Sentiment Analysis and Subjectivity. To appear in *Handbook of Natural Language Processing*, Second Edition, (editors: N. Indurkhya and F. J. Damerau), 2009 or 2010.

Y. Lu, and C. X. Zhai. 2008. Opinion integration through semi-supervised topic modeling. WWW-2008.

R. McDonald, K. Hannan, T. Neylon, M. Wells, and J. Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. ACL-2007

Q. Mei, X. Ling, M. Wondra, H. Su, and C. X. Zhai. 2007. Topic Sentiment Mixture: Modeling Facets and Opinions in Weblogs. WWW-2007.

V. Ng, S. Dasgupta, and S. M. Niaz Arifin. 2006. Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. ACL-2006.

B. Pang and L. Lee. 2008. Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval 2(1-2), pp. 1–135, 2008.

B. Pang, L. Lee. and S. Vaithyanathan. 2002. Thumbs up? Sentiment Classification Using Machine Learning Techniques. EMNLP-2002.

A-M. Popescu, and O. Etzioni. 2005. Extracting Product Features and Opinions from Reviews. EMNLP-2005.

G. Qiu, B. Liu, J. Bu and C. Chen. 2009. Expanding Domain Sentiment Lexicon through Double Propagation. IJCAI-2009.

E. Riloff, and J. Wiebe. 2003. Learning extraction patterns for subjective expressions. EMNLP-2003.

V. Stoyanov, and C. Cardie. 2008. Topic Identification for fine-grained opinion analysis. COLING-2008.

I. Titov and R. McDonald. 2008. A Joint Model of Text and Aspect Ratings for Sentiment Summarization. ACL-2008.

P. Turney. 2002. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. ACL-2002.

J. Wiebe, R. Bruce, and T. O'Hara. 1999. Development and use of a gold standard data set for subjectivity classifications. ACL-1999.

J. Wiebe, and T. Wilson. 2002. Learning to Disambiguate Potentially Subjective Expressions. CoNLL-2002.

T. Wilson, J. Wiebe. and R. Hwa. 2004. Just how mad are you? Finding strong and weak opinion clauses. AAAI-2004.

H. Yu, and Y. Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. EMNLP-2003.

Subjectivity Word Sense Disambiguation

Cem Akkaya and **Janyce Wiebe**
University of Pittsburgh
{cem,wiebe}@cs.pitt.edu

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

Abstract

This paper investigates a new task, *subjectivity word sense disambiguation (SWSD)*, which is to automatically determine which word instances in a corpus are being used with subjective senses, and which are being used with objective senses. We provide empirical evidence that SWSD is more feasible than full word sense disambiguation, and that it can be exploited to improve the performance of contextual subjectivity and sentiment analysis systems.

1 Introduction

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

Many approaches to opinion, sentiment, and subjectivity analysis rely on lexicons of words that may be used to express subjectivity. Examples of such words are the following (in bold):

(1) He is a **disease** to every team he has gone to.
Converting to SMF is a **headache**.
The concert left me **cold**.
That guy is such a **pain**.

Knowing the meaning (and thus subjectivity) of these words would help a system recognize the negative sentiments in these sentences.

Most subjectivity lexicons are compiled as lists of keywords, rather than word meanings (senses). However, many keywords have both subjective and objective senses. False hits – subjectivity clues used with objective senses – are a significant source of error in subjectivity and sentiment analysis. For example, even though the following sentence contains all of the negative keywords above, it is nevertheless objective, as they are all false hits:

(2) Early symptoms of the **disease** include severe **headaches**, red eyes, fevers and **cold** chills, body **pain**, and vomiting.

To tackle this source of error, we define a new task, *subjectivity word sense disambiguation (SWSD)*, which is to automatically determine which word instances in a corpus are being used with subjective senses, and which are being used with objective senses. We hypothesize that SWSD is more feasible than full word sense disambiguation, because it is more coarse grained – often, the exact sense need not be pinpointed. We also hypothesize that SWSD can be exploited to improve the performance of contextual subjectivity analysis systems via sense-aware classification.

The paper consists of two parts. In the first part, we build and evaluate a targeted supervised SWSD system that aims to disambiguate members of a subjectivity lexicon. It labels clue instances as having a subjective sense or an objective sense in context. The system relies on common machine learning features for word sense disambiguation (WSD). The performance is substantially above both baseline and the performance of full WSD on the same data, suggesting that the task is feasible, and that subjectivity provides a natural coarse-grained grouping of senses.

The second part demonstrates the promise of SWSD for contextual subjectivity analysis. First, we show that subjectivity sense ambiguity is highly prevalent in the MPQA opinion-annotated corpus (Wiebe et al., 2005; Wilson, 2008), thus establishing the potential benefit of performing SWSD. Then, we exploit SWSD to improve performance on several subjectivity analysis tasks, from subjective/objective sentence-level classification to positive/negative/neutral expression-level classification. To our knowledge, this is the

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 190–199,
Singapore, 6-7 August 2009.

first attempt to explicitly use sense-level subjectivity tags in contextual subjectivity and sentiment analysis.

2 Background

We adopt the definitions of *subjective* and *objective* from (Wiebe et al., 2005; Wiebe and Mihalcea, 2006; Wilson, 2008). Subjective expressions are words and phrases being used to express mental and emotional states, such as speculations, evaluations, sentiments, and beliefs. A general covering term for such states is *private state* (Quirk et al., 1985), an internal state that cannot be directly observed or verified by others. (Wiebe and Mihalcea, 2006) give the following examples:

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

Polarity (also called *semantic orientation*) is also important to NLP applications. In review mining, for example, we want to know whether an opinion about a product is positive or negative. Nonetheless, as argued by (Wiebe and Mihalcea, 2006; Su and Markert, 2008), there are also motivations for a separate subjective/objective (*S/O*) classification.

First, expressions may be subjective but not have any particular polarity. An example given by (Wilson et al., 2005a) is *Jerome says the hospital* **feels** *no different than a hospital in the states.* An NLP application system may want to find a wide range of private states attributed to a person, such as their motivations, thoughts, and speculations, in addition to their positive and negative sentiments. Second, benefits for sentiment analysis can be realized by decomposing the problem into *S*/*O* (or neutral versus polar) and polarity classification (Yu and Hatzivassiloglou, 2003; Pang and Lee, 2004; Wilson et al., 2005a; Kim and Hovy, 2006). We will see further evidence of this in Section 4.2.3 in this paper.

The contextual subjectivity analysis experiments in Section 4 include both *S*/*O* and polarity classifications. The data used in those experiments is from the MPQA Corpus (Wiebe et al., 2005; Wilson, 2008),[1] which consists of texts from the world press annotated for subjective expressions. In the MPQA Corpus, subjective expressions of varying lengths are marked, from single words to long phrases. In addition, other properties are annotated, including polarity.

For SWSD, we need the notions of subjective and objective *senses* of words in a dictionary. We adopt the definitions from (Wiebe and Mihalcea, 2006), who describe the annotation scheme as follows. Classifying a sense as *S* means that, when the sense is used in a text or conversation, one expects it to express subjectivity, and also that the phrase or sentence containing it expresses subjectivity. As noted in (Wiebe and Mihalcea, 2006), sentences containing objective senses may not be objective. Thus, objective senses are defined as follows: Classifying a sense as *O* means that, when the sense is used in a text or conversation, one does not expect it to express subjectivity and, if the phrase or sentence containing it is subjective, the subjectivity is due to something else. Finally, classifying a sense as *B* means it covers both subjective and objective usages.

The following subjective examples are given in (Wiebe and Mihalcea, 2006):

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

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

They give the following objective examples:

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

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

Wiebe and Mihalcea performed an agreement study and report that good agreement (κ=0.74) can be achieved between human annotators labeling the subjectivity of senses. For a similar task, (Su and Markert, 2008) also report good agreement (κ=0.79).

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

3 Subjectivity Word Sense Disambiguation

3.1 Task Definition and Method

We now turn to SWSD, and our method for performing it.

Note that SWSD is midway between pure dictionary classification and pure contextual interpretation. For SWSD, the context of the word is considered in order to *perform* the task, but the subjectivity is determined solely by the dictionary. In contrast, full contextual interpretation can deviate from a sense's subjectivity label in the dictionary. As noted above, words used with objective senses may appear in subjective expressions. For example, an SWSD system would label the following examples of alarm as *S*, *O* and *O*, respectively. On the other hand, a sentence-level subjectivity classifier would label the sentences as *S*, *S*, and *O*, respectively.

(4) His **alarm** grew.
Will someone shut that darn **alarm** off?
The **alarm** went off.

We use a supervised approach to SWSD. We train a different classifier for each lexicon entry for which we have training data. Thus, our approach is like targeted WSD (in contrast to all-words WSD), with two labels: *S* and *O*.

We borrow machine learning features which have been successfully used in WSD. Specifically, given an ambiguous target word, we use the following features from (Mihalcea, 2002):

CW : the target word itself
CP : POS of the target word
CF : surrounding context of 3 words and their POS
HNP : the head of the noun phrase to which the target word belongs
NB : the first noun before the target word
VB : the first verb before the target word
NA : the first noun after the target word
VA : the first verb after the target word
SK : at most 10 context words occurring at least 5 times; determined for each sense

3.2 Lexicon and Data

Our target words are members of a subjectivity lexicon, because, since they are in such a lexicon, we know they have subjective usages. Specifically, we use the lexicon of (Wilson et al., 2005b; Wilson, 2008).[2] The entries have been divided into those that are strongly subjective (*strongsubj*) and those that are weakly subjective (*weaksubj*), reflecting their reliability as subjectivity clues. The sources of the entries in the lexicon are identified in (Wilson, 2008). In the second part of this paper, we evaluate systems against the MPQA corpus. Wilson also uses this corpus for her evaluations. To enable this, entries were added to the lexicon independently from the MPQA corpus (that is, none of the entries were derived using the MPQA corpus).

The training and test data for SWSD consists of word instances in a corpus labeled as *S* or *O*, indicating whether they are used with a subjective or objective sense. Because we do not have data labeled with the *S*/*O* coarse-grained senses and we did not want to undertake the annotation effort at this stage, we created an annotated corpus by combining two types of sense annotations: (1) labels of senses within a dictionary as *S* or *O* (i.e., subjectivity sense labels), and (2) sense tags of word instances in a corpus (i.e., sense-tagged data). The subjectivity sense labels are used to collapse the sense labels in the sense-tagged data into the two new senses, *S* and *O*.

Our sense-tagged data are the lexical sample corpora (training and test data) from SENSEVAL1 (Kilgarriff and Palmer, 2000), SENSEVAL2 (Preiss and Yarowsky, 2001), and SENSEVAL3 (Mihalcea and Edmonds, 2004). We selected all of the SENSEVAL words that are also in the subjectivity lexicon, and labeled their dictionary senses as *S*, *O*, or *B* according to the annotation scheme described above in Section 2. We did this subjectivity sense labeling according to the sense inventory of the underlying corpus (Hector for SENSEVAL1; WordNet1.7 for SENSEVAL2; and WordNet1.7.1 for SENSEVAL3).

Among the words, we found that 11 are not ambiguous - either they have only *S* or only *O* senses (in the corresponding sense inventory), or the senses of their instances in the SENSEVAL data are all *S* or all *O*. So as not to inflate our results, we removed those 11 from the data, leaving 39 words. In addition, we excluded the senses labeled *B* (a total of 10 senses). This leaves a total of 372 senses: 9 words (64 senses) from SENSEVAL1, 18 words (201 senses) from SENSEVAL2, and 12 words (107 senses) from SENSEVAL3.

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

	Base	Acc	SP	SR	SF	OP	OR	OF	IB	EB(%)
All	79.9	88.3	89.3	89.1	89.2	87.1	87.4	87.2	8.4	41.8
S1	57.9	80.7	81.1	78.3	79.7	80.2	82.9	81.5	22.8	54.2
S2	81.1	87.3	86.5	85.2	85.8	87.9	89.0	88.4	6.2	32.8
S3	95.0	96.4	96.5	99.0	97.7	96.3	87.8	91.8	1.4	28.0

Table 1: Overall SWSD results (micro averages). *Base* is majority-class baseline; *Acc* is accuracy; *SP*, *SR*, and *SF* are subjective precision, recall and F-measure; similarly for *OP*, *OR*, and *OF*. *IB* is absolute improvement in Acc over Base; *EB* is percent error reduction in Acc.

3.3 SWSD Experiments

In this section, we evaluate our SWSD system, and compare its performance to an WSD system on the same data.

Note that, although generally in the SENSEVAL datasets, training and test data are provided separately, a few target words from SENSEVAL1 do not have both training and testing data. Thus, we opted to combine the training and test data into one dataset, and then perform 10-fold cross validation experiments.

For our classifier, we use the SVM classifier from the Weka package (Witten and Frank., 2005) with its default settings.

We were interested in how well the system would perform on more and less ambiguous words. Thus, we split the words into three subsets according to their majority-class baselines, and report separate results: *S1* (9 words), *S2* (18 words), and *S3* (12 words) have majority-class baselines in the intervals [50%,70%) , [70%,90%), and [90%,100%), respectively.

Table 1 contains the results, giving the overall results (micro averages), as well as results for the subsets *S1*, *S2*, and *S3*.

The improvement for SWSD over baseline is especially high for the less skewed set, *S1*. This is very encouraging because these words are the more ambiguous words, and thus are the ones that most need SWSD (assuming the SENSEVAL priors are similar to the priors in the corpus). The average error reduction over baseline for *S1* words is 54.2%. Even for the more skewed sets *S2* and *S3*, reductions are 32.8% and 28.0%, respectively, with an overall reduction of 41.8%.

To compare SWSD with WSD, we re-ran the 10-fold cross validation experiments, but this time using the original sense labels, rather than *S* and *O*. The (micro-averaged) accuracy is 67.9%, much lower than the overall accuracy for SWSD (88.3%).

The positive results provide evidence that SWSD is a feasible variant of WSD, and that the *S*/*O* sense groupings are natural ones, since the system is able to learn to distinguish between them with high accuracy. There is also potential for improvement by using a richer feature set, including subjectivity features.

4 Opinion Analysis with Subjectivity Word Sense Disambiguation

In this section, we explore the promise of SWSD for contextual subjectivity analysis. First, we provide evidence that a subjectivity lexicon can have substantial coverage of the subjective expressions in a corpus, yet still be responsible for significant subjectivity sense ambiguity in that corpus. Then, we exploit SWSD in several contextual opinion analysis systems, comparing the performance of sense-aware and non-sense-aware versions. They are all variations of components of the OpinionFinder opinion recognition system.[3]

4.1 Coverage and Ambiguity of Lexicon Entries in the MPQA Corpus

In this section, we consider the distribution of lexicon entries in the MPQA corpus.

The lexicon covers a substantial subset of the subjective expressions in the corpus: 67.1% of the subjective expressions contain one or more lexicon entries.

On the other hand, fully 42.9% of the instances of the lexicon entries in the MPQA corpus are not in subjective expressions. An instance that is not in a subjective expression is, by definition, being used with an objective sense. Thus, these instances are false hits of subjectivity clues. As mentioned above, the entries in the lexicon have been pre-classified as either more (*strongsubj*) or less (*weaksubj*) reliable. We see this difference reflected in their degree of ambiguity – 53% of the

[3] Available at http://www.cs.pitt.edu/opin

weaksubj instances are false hits, while only 22% of the *strongsubj* instances are.

The high coverage of the lexicon demonstrates its potential usefulness for opinion analysis systems, while its degree of ambiguity, in the form of false hits in a subjectivity annotated corpus, shows the potential benefit to opinion analysis of performing SWSD.

As mentioned above, our experiments involve only lexicon entries that are covered by the SENSEVAL data, as we did not perform manual sense tagging for this work. We have hope to expand the system's coverage in the future, as more word-sense tagged data is produced (e.g., ONTONOTES (Hovy et al., 2006)). We also have evidence that a moderate amount of manual annotation would be worth the effort. For example, let us order the lexicon entries from highest to lowest by frequency in the MPQA corpus. The top 20 are responsible for 25% of all false hits in the corpus; the top 40 are responsible for 34%; and the top 80 are responsible for 44%. If the SWSD system could be trained for these words, the potential impact on reducing false hits could be substantial, especially considering the good performance of the SWSD system on the more ambiguous words. Note that we do not want to simply discard these clues. The top 20 cover 9.4% of all subjective expressions; the top 40 cover 15.4%; and the top 80 cover 29.5%. Note that SWSD only needs the data annotated with the coarse-grained binary labels, which should be less time consuming to produce than full word sense tags.

4.2 Contextual Classification

We found in Section 3.3 that SWSD is a feasible task and then in Section 4.1 that there is a great deal of subjectivity sense ambiguity in a standard subjectivity-annotated corpus (MPQA). We now turn to exploiting the results of SWSD to automatically recognize subjectivity and sentiment in the MPQA corpus.

A motivation for using the MPQA data is that many types of classifiers have been evaluated on it, and we can directly test the effect of SWSD on these classifiers.

Note that, for the SWSD experiments, the number of words does not limit the amount of data, as SENSEVAL provides data for each word. However, the only parts of the MPQA corpus for which SWSD could affect performance is the subset containing instances of the words in the SWSD system's coverage. Thus, for the classifiers in this section, the data used is the *SenMPQA* dataset, which consists of the sentences in the MPQA Corpus that contain at least one instance of the 39 keywords. There are 689 such sentences (containing, in total, 723 instances of the 39 keywords).

Even though this dataset is smaller than the one used above, it gives us enough data to draw conclusions according to McNemar's test for statistical significance.

4.2.1 Rule-based Classifier

We first apply SWSD to the rule-based classifier from (Riloff and Wiebe, 2003). The classifier, which is a sentence-level *S/O* classifier, has low subjective and objective recall but high subjective and objective precision. It is useful for creating training data for subsequent processing by applying it to large amounts of unannotated data.

The classifier is a good candidate for directly measuring the effects of SWSD on contextual subjectivity analysis, because it classifies sentences only by looking for the presence of subjectivity keywords. Performance will improve if false hits can be ignored.

The classifier labels a sentence as *S* if it contains two or more *strongsubj* clues. On the other hand, it considers three conditions to classify a sentence as *O*: there are no *strongsubj* clues in the current sentence, there are together at most one *strongsubj* clue in the previous and next sentence, and there are together at most 2 *weaksubj* clues in the current, previous, and next sentence. A sentence that is not labeled *S* or *O* is labeled *unknown*.

The rule-based classifier is made sense aware by making it blind to the target word instances labeled *O* by the SWSD system, as these represent false hits of subjectivity keywords. We compare this sense-aware method (*SE*), with the original classifier (O_{RB}), in order to see if SWSD would improve performance. We also built another modified rule-based classifier *RE* to demonstrate the effect of randomly ignoring subjectivity keywords. *RE* ignores a keyword instance randomly with a probability of 0.429, the expected value of false hits in the MPQA corpus. The results are listed in Table 2.

The rule-based classifier looks for the presence of the keywords to find subjective sentences and for the absence of the keywords to find objective sentences. It is obvious that a variant working on

	Acc	OP	OR	OF	SP	SR	SF
O_{RB}	27.0	50.0	4.1	7.6	92.7	36.0	51.8
SE	28.3	62.1	9.3	16.1	92.7	35.8	51.6
RE	27.6	48.4	7.7	13.3	92.6	35.4	51.2

Table 2: Effect of SWSD on the rule-based classifiers.

fewer keyword instances than O_{RB} will always have the same or higher objective recall and the same or lower subjective recall than O_{RB}. That is the case for both *SE* and *RE*. The real benefit we see is in objective precision, which is substantially higher for *SE* than O_{RB}. For our experiments, *OP* gives a better idea of the impact of SWSD, because most of the keyword instances SWSD disambiguates are *weaksubj* clues, and *weaksubj* keywords figure more prominently in objective classification. On the other hand, *RE* has both lower *OP* and *SP* than O_{RB}. Note that accuracy for all three systems is low, because all *unknown* predictions are counted as incorrect.

These findings suggest that SWSD performs well on disambiguating keyword instances in the MPQA corpus,[4] and demonstrates a positive impact of SWSD on sentence-level subjectivity classification.

4.2.2 Subjective/Objective Classifier

We now move to more fine-grained expression-level subjectivity classification. Since sentences often contain multiple subjective expressions, expression-level classification is more informative than sentence-level classification.

The classifier in this section is an implementation of the *neutral/polar* supervised classifier of (Wilson et al., 2005a) (using the same features), except that the classes are *S*/*O* rather than *neutral/polar*. These classifiers label instances of lexicon entries. The gold standard is defined on the MPQA Corpus as follows: If an instance is in a subjective expression, it is contextually *S*. If the instance is in an objective expression, it is contextually *O*. We evaluate the system on the 723 clue instances in the SenMPQA dataset.

We incorporate SWSD information into the contextual subjectivity classifier in a straightforward fashion: outputs are modified according to simple, intuitive rules.

[4]which we cannot evaluate directly, as the MPQA corpus is not sense tagged.

Our strategy is defined by the relation between sense subjectivity and contextual subjectivity and involves two rules, *R1* and *R2*.

We know that a keyword instance used with a *S* sense must be in a subjective expression. *R1* is to simply trust SWSD: If the contextual classifier labels an instance as *O*, but SWSD determines that it has an *S* sense, then *R1* flips the contextual classifier's label to *S*.

Things are not as simple in the case of *O* senses, since they may appear in both subjective and objective expressions. We will state *R2*, and then explain it: If the contextual classifier labels an instance as *S*, but (1) SWSD determines that it has an *O* sense, (2) the contextual classifier's confidence is low, and (3) there is no other subjective keyword in the same expression, then *R2* flips the contextual classifier's label to *O*. First, consider confidence: though a keyword with an *O* sense may appear in either subjective or objective expressions, it is more likely to appear in an objective expression. We assume that this is reflected to some extent in the contextual classifier's confidence. Second, if a keyword with an *O* sense appears in a subjective expression, then the subjectivity is not due to that keyword but rather due to something else. Thus, the presence of another lexicon entry "explains away" the presence of the *O* sense in the subjective expression, and we do not want SWSD to overrule the contextual classifier. Only when the contextual classifier isn't certain and only when there isn't another keyword does *R2* flip the label to *O*.

Our definition of low confidence is in terms of the label weights assigned by BoosTexter (Schapire and Singer, 2000), which is the underlying machine learning algorithm of the classifier. We use the difference between the largest label weight and the second largest label weight as a measure of confidence, as suggested in the BoosTexter documentation. The threshold we use is 0.0008.[5]

We apply the contextual classifier and the SWSD system to the data, and compare the performance of the original system ($O_{S/O}$) and three sense-aware variants: one using only *R1*, one us-

[5]As will be noted below, we experimented with three thresholds for the classifier in Section 4.2.3, with no significant difference in accuracy. Here, we simply adopt 0.0008, without further experimentation. In addition, we did not experiment with other conditions than those incorporated in the two rules in this section and the two rules in Section 4.2.3 below.

	Acc	OP	OR	OF	SP	SR	SF
$O_{S/O}$	75.4	68.0	62.9	65.4	79.2	82.7	80.9
R1	77.7	75.5	58.8	66.1	78.6	88.8	83.4
R2	79.0	67.3	83.9	74.7	89.0	76.1	82.0
R1R2	81.3	72.5	79.8	75.9	87.4	82.2	84.8

Table 3: Effect of SWSD on the subjective/objective classifier

ing only *R2*, and one using both (*R1R2*). The results are in Table 3. The *R1* variant shows an improvement of 2.3 points in accuracy (a 9.4% error reduction). The *R2* variant shows an improvement of 3.6 points in accuracy (a 14.6% error reduction). Applying both rules (*R1R2*) gives an improvement of 5.9 percentage points in accuracy (a 24% error reduction).

In our case, a paired t-test is not appropriate to measure statistical significance, as we are not doing multiple runs. Thus, we apply McNemar's test, which is a non-parametric method for algorithms that can be executed only once, meaning training once and testing once (Dietterich, 1998). For *R1*, the improvement in accuracy is statistically significant at the $p < .05$ level. For *R2* and *R1R2*, the improvement in accuracy is statistically significant at the $p < .01$ level. Moreover, in all cases, we see improvement in both objective and subjective F-measure.

4.2.3 Contextual Polarity Classifier

We now apply SWSD to contextual polarity classification (positive/negative/neutral), in the hope that avoiding false hits of subjectivity keywords will also lead to performance improvement in contextual sentiment analysis.

We use an implementation of the classifier of (Wilson et al., 2005a). This classifier labels instances of lexicon entries. The gold standard is defined on the MPQA Corpus as follows: If an instance is in a positive subjective expression, it is contextually positive (*Ps*); if in a negative subjective expression, it is contextually negative (*Ng*); and if it is in an objective expression or a neutral subjective expression, then it is contextually *N(eutral)*. As above, we evaluate the system on the keyword instances in the SenMPQA dataset.

Wilson et al. use a two step approach. The first step classifies keyword instances as being in a polar (positive or negative) or a neutral context. The first step is performed by the neutral/polar classifier mentioned above in Section 4.2.2. The second step decides the contextual polarity (positive or negative) of the instances classified as polar in the first step, and is performed by a separate classifier.

To make a sense-aware version of the system, we use rules to change some of the answers of the neutral/polar classifier.

Unfortunately, we cannot simply trust SWSD when it labels a keyword as an *S* sense, because an *S* sense might be in a *N(eutral)* expression (since there are neutral subjective expressions). But, an *S* sense is more likely to appear in a *P(olar)* expression. Thus, we consider confidence (rule *R3*): If the contextual classifier labels an instance as *N*, but SWSD determines it has an *S* sense and the contextual classifier's confidence is low,[6] then *R3* flips the contextual classifier's label to *P*.

Rule *R4* is analogous to *R2* in the previous section: If the contextual classifier labels an instance as *P*, but (1) SWSD determines that it has an *O* sense, (2) the contextual classifier's confidence is low, and (3) there is no other subjective keyword in the same expression, then *R2* flips the contextual classifier's label to *N*.

We compare the performance of the original neutral/polar classifier ($O_{N/P}$) and sense-aware variants using *R3* and *R4*. The results are in Table 4. This time, the table does not include a combined method, because only *R4* improves performance. This is consistent with the finding in (Wilson et al., 2005a) that most errors are caused by subjectivity keywords with non-neutral prior polarity appearing in phrases with neutral contextual polarity. *R4* targets these cases. It is promising to see that SWSD provides enough information to fix some of them. There is a 2.6 point improvement in accuracy (a 12.4% error reduction). The improvement in accuracy is statistically significant at the $p < .01$ level with McNemar's test. The improvement in accuracy is accompanied by improvements in both neutral and polar F-measure.

We wanted to see if the improvements in the

[6] As in the previous section, low confidence is defined in terms of the difference between the largest label weight and the second largest label weight assigned by BoosTexter. We tried three thresholds, 0.0007, 0.0008, and 0.0009, resulting in only a slight difference in accuracy: 0.0007 and 0.0009 both give 81.5 accuracy compared to 81.6 accuracy for 0.0008. We report results using 0.0008, though the accuracy using the other thresholds is statistically significantly better than the accuracy of the original classifier at the same level.

	Acc	NP	NR	NF	NgP	NgR	NgF	PsP	PsR	PsF
$O_{Ps/Ng/N}$	77.6	80.9	94.6	87.2	60.4	29.4	39.5	52.2	32.4	40.0
R4	80.6	81.2	98.7	89.1	82.1	29.4	43.2	68.6	32.4	44.0

Table 5: Effect of SWSD on the contextual polarity classifier

	Acc	NP	NR	NF	PP	PR	PF
$O_{N/P}$	79.0	81.5	92.5	86.7	65.8	40.7	50.3
R3	70.0	83.7	73.8	78.4	44.4	59.3	50.8
R4	81.6	81.7	96.8	88.6	81.1	38.6	52.3

Table 4: Effect of SWSD on the neutral/polar classifier

first step of Wilson et al's system can be propagated to their second step, yielding an overall improvement in positive /negative/neutral (*Ps*/*Ng*/*N*) classification.

The sense-aware variant of the overall two-part system is the same as the original except that we apply *R4* to the output of the first step (flipping some of the neutral/polar classifier's *P* labels to *N*). Thus, since the second step in Wilson et al.'s classifier processes only those instances labeled *P* in the first step, in the sense-aware system, fewer instances are passed from the first to the second step.

Table 5 reports results for the original system ($O_{Ps/Ng/N}$) and the sense-aware variant (*R4*). These results are for the entire SenMPQA dataset, not just those labeled *P* in the first step.

The accuracy improves 3 percentage points (a 13.4% error reduction). The improvement in accuracy is statistically significant at the $p < .01$ level with McNemar's test. We see the real benefit when we look at the precision of the positive and negative classes. Negative precision goes from 60.4 to 82.1 and positive precision goes from 52.2 to 68.6, with no loss in recall. This is evidence that the SWSD system is doing a good job of removing some false hits of subjectivity clues that harm the original version of the system.

5 Comparisons to Previous Work

Several researchers exploit lexical resources for contextual subjectivity and sentiment analysis. These systems typically look for the presence of subjective or sentiment-bearing words in the text. They may rely only on this information (e.g., (Turney, 2002; Whitelaw et al., 2005; Riloff and Wiebe, 2003)), or they may combine it with additional information as well (e.g., (Yu and Hatzivassiloglou, 2003; Kim and Hovy, 2004; Bloom et al., 2007; Wilson et al., 2005a)). We apply SWSD to some of those systems to show the effect of SWSD on contextual subjectivity and sentiment analysis.

Another set of related work is on subjectivity and polarity labeling of word senses (e.g. (Esuli and Sebastiani, 2006; Andreevskaia and Bergler, 2006; Wiebe and Mihalcea, 2006; Su and Markert, 2008)). They label senses of words in a dictionary. In comparison, we label senses of word instances in a corpus.

Moreover, our work extends findings in (Wiebe and Mihalcea, 2006) and (Su and Markert, 2008). (Wiebe and Mihalcea, 2006) demonstrates that subjectivity is a property that can be associated with word senses. We show that it is a natural grouping of word senses and that it provides a principled way for clustering senses. They also demonstrate that subjectivity helps with WSD. We show that a coarse-grained WSD variant (SWSD) helps with subjectivity and sentiment analysis. Both (Wiebe and Mihalcea, 2006) and (Su and Markert, 2008) show that even reliable subjectivity clues have objective senses. We demonstrate that this ambiguity is also prevalent in a corpus.

Several researchers (e.g., (Palmer et al., 2004; Navigli, 2006; Snow et al., 2007; Hovy et al., 2006)) work on reducing the granularity of sense inventories for WSD. They aim for a more coarse-grained sense inventory to overcome performance shortcomings related to fine-grained sense distinctions. Our work is similar in the sense that we reduce all senses of a word to two senses (*S*/*O*). The difference is the criterion driving the grouping. Related work concentrates on syntactic and semantic similarity between senses to group them. In contrast, our grouping is driven by subjectivity with a specific application area in mind, namely subjectivity and sentiment analysis.

6 Conclusions and Future Work

We introduced the task of subjectivity word sense disambiguation (SWSD), and evaluated a supervised method inspired by research in WSD. The

system achieves high accuracy, especially on highly ambiguous words, and substantially outperforms WSD on the same data. The positive results provide evidence that SWSD is a feasible variant of WSD, and that the *S*/*O* sense groupings are natural ones.

We also explored the promise of SWSD for contextual subjectivity analysis. We showed that a subjectivity lexicon can have substantial coverage of the subjective expressions in the corpus, yet still be responsible for significant sense ambiguity. This demonstrates the potential benefit to opinion analysis of performing SWSD. We then exploit SWSD in several contextual opinion analysis systems, including positive/negative/neutral sentiment classification. Improvements in performance were realized for all of the systems.

We plan several future directions which promise to further increase the impact of SWSD on subjectivity and sentiment analysis. We will manually annotate a moderate number of strategically chosen words, namely frequent ones which are highly ambiguous. In addition, we will add features to the SWSD system reflecting the subjectivity of the surrounding context. Finally, there are more sophisticated strategies to explore for improving subjectivity and sentiment analysis via SWSD than the simple, intuitive rules we began with in this paper.

Acknowledgments

This material is based in part upon work supported by National Science Foundation awards #0840632 and #0840608. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

A. Andreevskaia and S. Bergler. 2006. Mining wordnet for a fuzzy sentiment: Sentiment tag extraction from wordnet glosses. In *(EACL-2006)*.

K. Bloom, N. Garg, and S. Argamon. 2007. Extracting appraisal expressions. In *HLT-NAACL 2007*, pages 308–315, Rochester, NY.

T. G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923.

A. Esuli and F. Sebastiani. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *(LREC-06)*, Genova, IT.

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, New York City.

A. Kilgarriff and M. Palmer, editors. 2000. *Computer and the Humanities. Special issue: SENSEVAL. Evaluating Word Sense Disambiguation programs*, volume 34, April.

S.-M. Kim and E. Hovy. 2004. Determining the sentiment of opinions. In *(COLING 2004)*, pages 1267–1373, Geneva, Switzerland.

S.-M. Kim and E. Hovy. 2006. Identifying and analyzing judgment opinions. In *(HLT/NAACL-06)*, pages 200–207, New York, New York.

R. Mihalcea and P. Edmonds, editors. 2004. *Proceedings of SENSEVAL-3, Association for Computational Linguistics Workshop*, Barcelona, Spain.

R. Mihalcea. 2002. Instance based learning with automatic feature selection applied to Word Sense Disambiguation. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan, August.

R. Navigli. 2006. Meaningful clustering of senses helps boost word sense disambiguation performance. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia.

M. Palmer, O. Babko-Malaya, and H. T. Dang. 2004. Different sense granularities for different applications. In *HLT-NAACL 2004 Workshop: 2nd Workshop on Scalable Natural Language Understanding*, Boston, Massachusetts.

B. Pang and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *(ACL-04)*, pages 271–278, Barcelona, ES. Association for Computational Linguistics.

J. Preiss and D. Yarowsky, editors. 2001. *Proceedings of SENSEVAL-2, Association for Computational Linguistics Workshop*, Toulouse, France.

R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. 1985. *A Comprehensive Grammar of the English Language.* Longman, New York.

E. Riloff and J. Wiebe. 2003. Learning extraction patterns for subjective expressions. In *(EMNLP-2003)*, pages 105–112, Sapporo, Japan.

R. E. Schapire and Y. Singer. 2000. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.

R. Snow, S. Prakash, D. Jurafsky, and A. Ng. 2007. Learning to merge word senses. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic.

F. Su and K. Markert. 2008. From word to sense: a case study of subjectivity recognition. In *(COLING-2008)*, Manchester.

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

C. Whitelaw, N. Garg, and S. Argamon. 2005. Using appraisal groups for sentiment analysis. In *Proceedings of CIKM-05, the ACM SIGIR Conference on Information and Knowledge Management*, Bremen, DE.

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

J. Wiebe, T. Wilson, and C. Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation (formerly Computers and the Humanities)*, 39(2/3):164–210.

T. Wilson, J. Wiebe, and P. Hoffmann. 2005a. Recognizing contextual polarity in phrase-level sentiment analysis. In *(HLT/EMNLP-2005)*, pages 347–354, Vancouver, Canada.

T. Wilson, P. Hoffmann, S. Somasundaran, J. Kessler, J. Wiebe, Y. Choi, C. Cardie, E. Riloff, and S. Patwardhan. 2005b. OpinionFinder: A system for subjectivity analysis. In *Proc. Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-2005) Companion Volume (software demonstration)*.

T. Wilson. 2008. *Fine-grained Subjectivity and Sentiment Analysis: Recognizing the Intensity, Polarity, and Attitudes of private states*. Ph.D. thesis, Intelligent Systems Program, University of Pittsburgh.

I. Witten and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann, June.

H. Yu and V. Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-03)*, pages 129–136, Sapporo, Japan.

Non-Projective Parsing for Statistical Machine Translation

Xavier Carreras **Michael Collins**
MIT CSAIL, Cambridge, MA 02139, USA
{carreras,mcollins}@csail.mit.edu

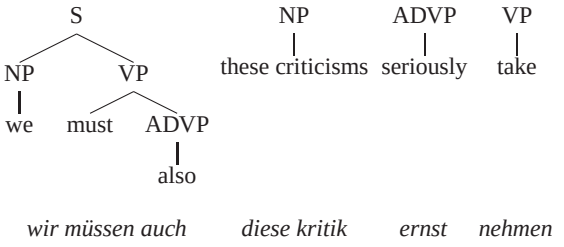
Abstract

We describe a novel approach for syntax-based statistical MT, which builds on a variant of tree adjoining grammar (TAG). Inspired by work in discriminative dependency parsing, the key idea in our approach is to allow highly flexible reordering operations during parsing, in combination with a discriminative model that can condition on rich features of the source-language string. Experiments on translation from German to English show improvements over phrase-based systems, both in terms of BLEU scores and in human evaluations.

1 Introduction

Syntax-based models for statistical machine translation (SMT) have recently shown impressive results; many such approaches are based on either synchronous grammars (e.g., (Chiang, 2005)), or tree transducers (e.g., (Marcu et al., 2006)). This paper describes an alternative approach for syntax-based SMT, which directly leverages methods from non-projective dependency parsing. The key idea in our approach is to allow highly flexible reordering operations, in combination with a discriminative model that can condition on rich features of the source-language input string.

Our approach builds on a variant of tree adjoining grammar (TAG; (Joshi and Schabes, 1997)) (specifically, the formalism of (Carreras et al., 2008)). The models we describe make use of phrasal entries augmented with subtrees that provide syntactic information in the target language. As one example, when translating the sentence *wir müssen auch diese kritik ernst nehmen* from German into English, the following sequence of syntactic phrasal entries might be used (we show each English syntactic fragment above its associated German sub-string):



TAG parsing operations are then used to combine these fragments into a full parse tree, giving the final English translation *we must also take these criticisms seriously*.

Some key aspects of our approach are as follows:

• We impose no constraints on entries in the phrasal lexicon. The method thereby retains the full set of lexical entries of phrase-based systems (e.g., (Koehn et al., 2003)).[1]

• The model allows a straightforward integration of lexicalized syntactic language models—for example the models of (Charniak, 2001)—in addition to a surface language model.

• The operations used to combine tree fragments into a complete parse tree are significant generalizations of standard parsing operations found in TAG; specifically, they are modified to be highly flexible, potentially allowing any possible permutation (reordering) of the initial fragments.

As one example of the type of parsing operations that we will consider, we might allow the tree fragments shown above for *these criticisms* and *take* to be combined to form a new structure with the sub-string *take these criticisms*. This step in the derivation is necessary to achieve the correct English word order, and is novel in a couple of respects: first, *these criticisms* is initially seen to the left of *take*, but after the adjunction this order is reversed; second, and more unusually, the treelet for *seriously* has been skipped over, with the result that the German words translated at this point (*diese*, *kritik*, and *nehmen*) form a non-contiguous sequence. More generally, we will allow any two

[1] Note that in the above example each English phrase consists of a completely connected syntactic structure; this is not, however, a required constraint, see section 3.2 for discussion.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 200–209,
Singapore, 6-7 August 2009.

tree fragments to be combined during the translation process, irrespective of the reorderings which are introduced, or the non-projectivity of the parsing operations that are required.

The use of flexible parsing operations raises two challenges that will be a major focus of this paper. First, these operations will allow the model to capture complex reordering phenomena, but will in addition introduce many spurious possibilities. Inspired by work in discriminative dependency parsing (e.g., (McDonald et al., 2005)), we add probabilistic constraints to the model through a discriminative model that links lexical dependencies in the target language to features of the source language string. We also investigate hard constraints on the dependency structures that are created during parsing. Second, there is a need to develop efficient decoding algorithms for the models. We describe approximate search methods that involve a significant extension of decoding algorithms originally developed for phrase-based translation systems.

Experiments on translation from German to English show a 0.5% improvement in BLEU score over a phrase-based system. Human evaluations show that the syntax-based system gives a significant improvement over the phrase-based system. The discriminative dependency model gives a 1.5% BLEU point improvement over a basic model that does not condition on the source language string; the hard constraints on dependency structures give a 0.8% BLEU improvement.

2 Relationship to Previous Work

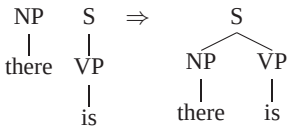
A number of syntax-based translation systems have framed translation as a parsing problem, where search for the most probable translation is achieved using algorithms that are generalizations of conventional parsing methods. Early examples of this work include (Alshawi, 1996; Wu, 1997); more recent models include (Yamada and Knight, 2001; Eisner, 2003; Melamed, 2004; Zhang and Gildea, 2005; Chiang, 2005; Quirk et al., 2005; Marcu et al., 2006; Zollmann and Venugopal, 2006; Nesson et al., 2006; Cherry, 2008; Mi et al., 2008; Shen et al., 2008). The majority of these methods make use of synchronous grammars, or tree transducers, which operate over parse trees in the source and/or target languages. Reordering rules are typically specified through rotations or transductions stated at the level of context-free rules, or larger fragments, within parse trees. These rules can be learned automatically from corpora.

A critical difference in our work is to allow arbitrary reorderings of the source language sentence (as in phrase-based systems), through the use of flexible parsing operations. Rather than stating reordering rules at the level of source or target language parse trees, we capture reordering phenomena using a discriminative dependency model. Other factors that distinguish us from previous work are the use of all phrases proposed by a phrase-based system, and the use of a dependency language model that also incorporates constituent information (although see (Charniak et al., 2003; Shen et al., 2008) for related approaches).

3 A Syntactic Translation Model

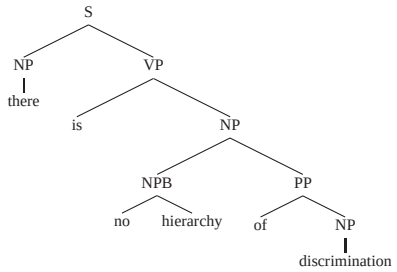
3.1 Background

Our work builds on the variant of tree adjoining grammar (TAG) introduced by (Carreras et al., 2008). In this formalism the basic units in the grammar are spines, which associate tree fragments with lexical items. These spines can be combined using a *sister-adjunction* operation (Rambow et al., 1995), to form larger pieces of structure.[2] For example, we might have the following operation:



In this case the spine for *there* has sister-adjoined into the S node in the spine for *is*; we refer to the spine for *there* as being the modifier spine, and the spine for *is* being the head spine. There are close connections to dependency formalisms: in particular in this operation we see a lexical dependency between the modifier word *there* and the head word *is*. It is possible to define syntactic language models, similar to (Charniak, 2001), which associate probabilities with these dependencies, roughly speaking of the form $P(w_m, s_m | w_h, s_h, pos, \sigma)$, where w_m and s_m are the identities of the modifier word and spine, w_h and s_h are the identities of the head word and spine, pos is the position in the head spine that is being adjoined into, and σ is some additional state (e.g., state that tracks previous modifiers that have adjoined into the same spine).

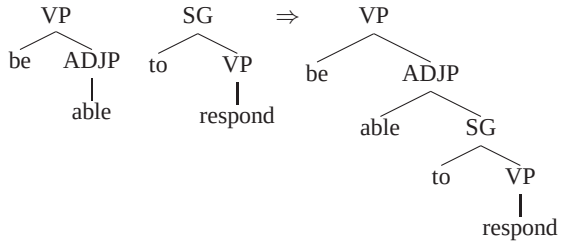
[2] We also make use of the r-adjunction operation defined in (Carreras et al., 2008), which, together with sister-adjunction, allows us to model the full range of structures found in the Penn treebank.



es gibt keine hierarchie der diskriminierung

Figure 1: A training example consisting of an English (target language) tree and a German (source language) sentence.

In this paper we will also consider *treelets*, which are a generalization of spines, and which allow lexical entries that include more than one word. These treelets can again be combined using a sister-adjunction operation. As an example, consider the following operation:



In this case the treelet for *to respond* sister-adjoins into the treelet for *be able*. This operation introduces a bi-lexical dependency between the modifier word *to* and the head word *able*.

3.2 S-phrases

This section describes how phrase entries from phrase-based translation systems can be modified to include associated English syntactic structures. These syntactic phrase-entries (from here on referred to as "s-phrases") will form the basis of the translation models that we describe.

We extract s-phrases from training examples consisting of a source-language string paired with a target-language parse tree. For example, consider the training example in figure 1. We assume some method that enumerates a set of possible *phrase entries* for each training example: each phrase entry is a pair $\langle (i,j),(k,l) \rangle$ specifying that source-language words $f_i \ldots f_j$ correspond to target-language words $e_k \ldots e_l$ in the example. For example, one phrase entry for the example might be $\langle (1,2),(1,2) \rangle$, representing the pair $\langle$*es gibt* $\Rightarrow$ *there is*$\rangle$. In our experiments we use standard methods in phrase-based systems (Koehn et al., 2003) to define the set of phrase entries for each sentence in training data.

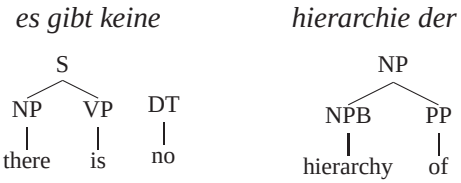
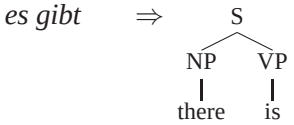


Figure 2: Example syntactic phrase entries. We show German sub-strings above their associated sequence of treelets.[4]

For each phrase entry, we add syntactic information to the English string. To continue our example, the resulting entry would be as follows:



To give a more formal description of how syntactic structures are derived for phrases, first note that each parse tree t is mapped to a TAG derivation using the method described in (Carreras et al., 2008). This procedure uses the head finding rules of (Collins, 1997). The resulting derivation consists of a TAG spine for each word seen in the sentence, together with a set of adjunction operations which each involve a modifier spine and a head spine. Given an English string $\mathbf{e} = e_1 \ldots e_n$, with an associated parse tree t, the syntactic structure associated with a substring $e_k \ldots e_l$ (e.g., *there is*) is then defined as follows:

• For each word in the English sub-string, include its associated TAG spine in t.

• In addition, include any adjunction operations in t where both the head and modifier word are in the sub-string $e_j \ldots e_k$.

In the above example, the resulting structure (i.e., the structure for *there is*) is a single treelet. In other cases, however, we may get a sequence of treelets, which are disconnected from each other. For example, another likely phrase-entry for this training example is $\langle$*es gibt keine* $\Rightarrow$ *there is no*$\rangle$ resulting in the first lexical entry in figure 2, which has two treelets. Allowing s-phrases with multiple treelets ensures that all phrases used by phrase-based systems can be used within our approach.

As a final step, we add additional *alignment* information to each s-phrase. Consider an s-phrase which contains source-language words $f_1 \ldots f_n$ paired with target-language words $e_1 \ldots e_m$. The alignment information is a vector $\langle (a_1,b_1) \ldots (a_m,b_m) \rangle$ that specifies for each word e_i its alignment to words $f_{a_i} \ldots f_{b_i}$ in the source language. For example, for the phrase en-

try $\langle$*es gibt* $\Rightarrow$ *there is*$\rangle$ a correct alignment would be $\langle(1,1),(2,2)\rangle$, specifying that *there* is aligned to *es*, and *is* is aligned to *gibt* (note that in many, but not all, cases $a_i = b_i$, i.e., a target language word is aligned to a single source language word).

The alignment information in s-phrases will be useful in tying syntactic dependencies created in the target language to positions in the source language string. In particular, we will consider discriminative models (analogous to models for dependency parsing, e.g., see (McDonald et al., 2005)) that estimate the probability of target-language dependencies conditioned on properties of the source-language string. Alignments may be derived in a number of ways; in our method we directly use phrase entries proposed by a phrase-based system. Specifically, for each target word e_i in a phrase entry $\langle f_1 \ldots f_n, e_1 \ldots e_m \rangle$ for a training example, we find the smallest[5] phrase entry in the same training example that includes e_i on the target side, and is a subset of $f_1 \ldots f_n$ on the source side; the word e_i is then aligned to the subset of source language words in this "minimal" phrase.

In conclusion, s-phrases are defined as follows:

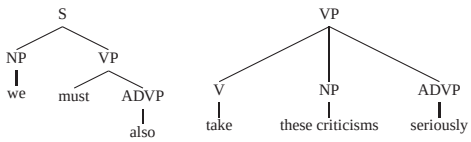
Definition 1 *An s-phrase is a 4-tuple $\langle f, e, t, a \rangle$ where: f is a sequence of foreign words; e is a sequence of English words; t is a sequence of treelets specifying a TAG spine for each English word, and potentially some adjunctions between these spines; and a is an alignment. For an s-phrase q we will sometimes refer to the 4 elements of q as $f(q)$, $e(q)$, $t(q)$ and $a(q)$.*

3.3 The Model

We now introduce a model that makes use of s-phrases, and which is flexible in the reorderings that it allows. To provide some intuition, and some motivation for the use of reordering operations, figure 3 gives several examples of German strings which have different word orders from English.

The crucial idea will be to use TAG adjunction operations to combine treelets to form a complete parse tree, but with a complete relaxation on the order in which the treelets are combined. For example, consider again the example given in the introduction to this paper. In the first step of a derivation that builds on these treelets, the treelet for *these criticisms* might adjoin into the treelet for *take*, giving the following new sequence:

In the next derivation step *seriously* is adjoined to the right of *take*, giving the following treelets:

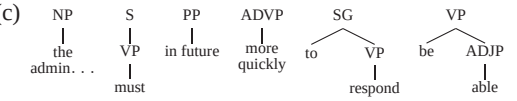


In the final step the second treelet adjoins into the VP above *must*, giving a parse tree for the string *we must also take these criticisms seriously*, and completing the translation.

Formally, given an input sentence $\mathbf{f}$, a derivation d is a pair $\langle \mathbf{q}, \pi \rangle$ where:

- $\mathbf{q} = q_1 \ldots q_n$ is a sequence of s-phrases such that $\mathbf{f} = f(q_1) \oplus f(q_2) \oplus \ldots \oplus f(q_n)$ (where $u \oplus v$ denotes the concatenation of strings u and v).
- π is a set of adjunction operations that connects the sequence of treelets contained in $\langle t(q_1), t(q_2), \ldots, t(q_n) \rangle$ into a parse tree in the target language. The operations allow a complete relaxation of word order, potentially allowing any of the $n!$ possible orderings of the n s-phrases. We make use of both sister-adjunction and r-adjunction operations, as defined in (Carreras et al., 2008).[6]

1(a) [die verwaltung] [muss] [künftig] [schneller] [reagieren] [können] 1(b) the administration must be able to respond more quickly in future

1(c)


2(a) [meiner ansicht nach] [darf] [der erweiterungsprozess] [nicht] [unnötig] [verzögert] [werden] 2(b) in my opinion the expansion process should not be delayed unnecessarily

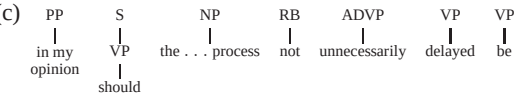
2(c)


Figure 3: Examples of translations. In each example (a) is the original German string, with a possible segmentation marked with "[" and "]"; (b) is a translation for (a); and (c) is a sequence of phrase entries, including syntactic structures, for the segmentation given in (a).

[5] The "size" of a phrase entry is defined to be $n_s + n_t$ where n_s is the number of source language words in the phrase, n_t is the number of target language words.

[6] In principle we allow any treelet to adjoin into any other treelet—for example there are no hard, grammar-based constraints ruling out the combination of certain pairs of non-terminals. Note however that in some cases operations will have probability 0 under the syntactic language model introduced later in this section.

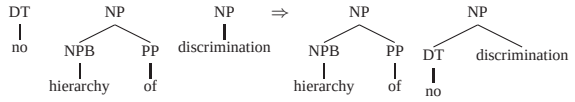


Figure 4: A spurious derivation step. The treelets arise from *[keine] [hierarchie der] [diskriminierung]*.

Given a derivation $d = \langle \mathbf{q}, \pi \rangle$, we define $e(d)$ to be the target-language string defined by the derivation, and $t(d)$ to be the complete target-language parse tree created by the derivation. The most likely derivation for a foreign sentence $\mathbf{f}$ is $\arg\max_{d \in G(\mathbf{f})} score(d)$, where $G(\mathbf{f})$ is the set of possible derivations for $\mathbf{f}$, and the score for a derivation is defined as[7]

$$\begin{aligned} score(d) &= score_{LM}(e(d)) + score_{SYN}(t(d)) \\ &+ score_R(d) + \sum_{j=1}^{n} score_P(q_j) \end{aligned} \quad (1)$$

The components of the model are as follows:

- $score_{LM}(e(d))$ is the log probability of the English string under a trigram language model.
- $score_{SYN}(t(d))$ is the log probability of the English parse tree under a syntactic language model, similar to (Charniak, 2001), that associates probabilities with lexical dependencies.
- $score_R(d)$ will be used to score the parsing operations in π, based on the source-language string and the alignments in the s-phrases. This part of the model is described extensively in section 4.1 of this paper.
- $score_P(q)$ is the score for an s-phrase q. This score is a log-linear combination of various features, including features that are commonly found in phrase-based systems: for example $\log P(f(q)|e(q))$, $\log P(e(q)|f(q))$, and lexical translation probabilities. In addition, we include a feature $\log P(t(q)|f(q), e(q))$, which captures the probability of the phrase in question having the syntactic structure $t(q)$.

Note that a model that includes the terms $score_{LM}(e(d))$ and $\sum_{j=1}^{n} score_P(q_j)$ alone would essentially be a basic phrase-based model (with no distortion terms). The terms $score_{SYN}(t(d))$ and $score_R(d)$ add syntactic information to this basic model.

A key motivation for this model is the flexibility of the reordering operations that it allows. However, the approach raises two major challenges:

[7] In practice, MERT training (Och, 2003) will be used to train relative weights for the different model components.

Constraints on reorderings. Relaxing the operations in the parsing model will allow complex reorderings to be captured, but will also introduce many spurious possibilities. As one example, consider the derivation step shown in figure 4. This step may receive a high probability from a syntactic or surface language model—*no discrimination* is a quite plausible `NP` in English—but it should be ruled out for other reasons, for example because it does not respect the dependencies in the original German (i.e., *keine*/no is not a modifier to *diskriminierung*/discrimination in the German string). The challenge will be to develop either hard constraints which rule out spurious derivation steps such as these, or soft constraints, encapsulated in $score_R(d)$, which penalize them.

Efficient search. Exact search for the derivation which maximizes the score in Eq. 1 cannot be accomplished efficiently using dynamic programming (as in phrase-based systems, it is easy to show that the decoding problem is NP-complete). Approximate search methods will be needed.

The next two sections of this paper describe solutions to these two challenges.

4 Constraints on Reorderings

4.1 A Discriminative Dependency Model

We now describe the model $score_R$ introduced in the previous section. Recall that π specifies k adjunction operations that are used to build a full parse tree, where $k \geq n$ is the number of treelets within the sequence of s-phrases $\mathbf{q} = \langle q_1 \ldots q_n \rangle$.

Each of the k adjunction operations creates a dependency between a modifier word w_m within a phrase q_m, and a head word w_h within a phrase q_h. For example, in the example in section 3.3 where *these criticisms* was combined with *take*, the modifier word is *criticisms* and the head word is *take*. The modifier and head words have TAG spines s_m and s_h respectively. In addition we can define (a_m, b_m) to be the start and end indices of the words in the foreign string to which the word w_m is aligned; this information can be recovered because the s-phrase q_m contains alignment information for all target words in the phrase, including w_m. Similarly, we can define (a_h, b_h) to be alignment information for the head word w_h. Finally, we can define ρ to be a binary flag specifying whether or not the adjunction operation involves reordering (in the *take criticism* example, this flag is set to `true`, because the order in En-

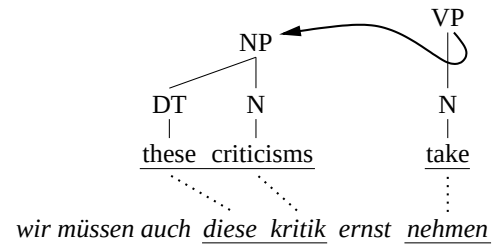


Figure 5: An adjunction operation that involves the modifier *criticisms* and the head *take*. The phrases involved are underlined; the dotted lines show alignments within s-phrases between English words and positions in the German string. The Γ-dependency in this case includes the head and modifier words, together with their spines, and their alignments to positions in the German string (*kritik* and *nehmen*).

glish is reversed from that in German). This leads to the following definition:

Definition 2 *Given a derivation $d = \langle \mathbf{q}, \pi \rangle$, we define $\Gamma(d)$ to be the set of* Γ-dependencies *in d. Each* Γ*-dependency is a tuple $\langle w_m, s_m, a_m, b_m, w_h, s_h, a_h, b_h, \rho \rangle$ of elements as described above.*

Figure 5 gives an illustration of how an adjunction creates one such Γ-dependency.

The model is then defined as

$$score_R(d) = \sum_{\gamma \in \Gamma(d)} score_r(\gamma, \mathbf{f})$$

where $score_r(\gamma, \mathbf{f})$ is a score associated with the Γ-dependency γ. This score can potentially be sensitive to any information in γ or the source-language string $\mathbf{f}$; in particular, note that the alignment indices (a_m, b_m) and (a_h, b_h) essentially anchor the target-language dependency to positions in the source-language string, allowing the score for the dependency to be based on features that have been widely used in discriminative dependency parsing, for example features based on the proximity of the two positions in the source-language string, the part-of-speech tags in the surrounding context, and so on. These features have been shown to be powerful in the context of regular dependency parsing, and our intent is to leverage them in the translation problem.

In our model, we define $score_r$ as follows. We estimate a model $P(y|\gamma, \mathbf{f})$ where $y \in \{-1, +1\}$, and $y = +1$ indicates that a dependency does exist between w_m and w_h, and $y = -1$ indicates that a dependency does not exist. We then define

$$score_r(\gamma, \mathbf{f}) = \log P(+1|\gamma, \mathbf{f})$$

To estimate $P(y|\gamma, \mathbf{f})$, we first extract a set of labeled training examples of the form $\langle y_i, \gamma_i, \mathbf{f}_i \rangle$ for $i = 1 \ldots N$ from our training data as follows: for each pair of target-language words (w_m, w_h) seen in the training data, we can extract associated spines (s_m, s_h) from the relevant parse tree, and also extract a label y indicating whether or not a head-modifier dependency is seen between the two words in the parse tree. Given an s-phrase in the training example that includes w_m, we can extract alignment information (a_m, b_m) from the s-phrase; we can extract similar information (a_h, b_h) for w_h. The end result is a training example of the form $\langle y, \gamma, \mathbf{f} \rangle$.[8] We then estimate $P(y|\gamma, \mathbf{f})$ using a simple backed-off model that takes into account the identity of the two spines, the value for the flag r, the distance between (a_m, b_m) and (a_h, b_h), and part-of-speech information in the source language.

4.2 Contiguity of π-Constituents

We now describe a second type of constraint, which limits the amount of non-projectivity in derivations. Consider again the k adjunction operations in π, which are used to connect treelets into a full parse tree. Each adjunction operation involves a head treelet that *dominates* a modifier treelet. Thus for any treelet t, we can consider its *descendants*, that is, the entire set of treelets that are directly or indirectly dominated by t. We define a *π-constituent* for treelet t to be the subset of source-language words dominated by t and its descendants. We then introduce the following constraint on π-constituents:

Definition 3 *(π-constituent constraint.) A* π*-constituent is* contiguous *iff it consists of a contiguous sequence of words in the source language. A derivation π satisfies the π-constituent constraint iff all π-constituents that it contains are contiguous.*

In this paper we constrain all derivations to satisfy the π-constituent constraint (future work may consider probabilistic versions of the constraint).

The intuition behind the constraint deserves more discussion. The constraint specifies that the modifiers to each treelet can appear in any order around the treelet, with arbitrary reorderings or non-projective operations. However, once a treelet has taken all its modifiers, the resulting π-constituent must form a contiguous sub-sequence

[8] To be precise, there may be multiple (or even zero) s-phrases which include w_m or w_h, and these s-phrases may include conflicting alignment information. Given n_m different alignments seen for w_m, and n_h different alignments seen for w_h, we create $n_m \times n_h$ training examples, which include all possible combinations of alignments.

of the source-language string. As one set of examples, consider the translations in figure 3, and the example given in the introduction. These examples involve reordering of arguments and adjuncts within clauses, a very common case of reordering in translation from German to English. The reorderings in these translations are quite flexible, but in all cases satisfy the π-constituent constraint.

As an illustration of a derivation that violates the constraint, consider again the derivation step shown in figure 4. This step has formed a partial hypothesis, *no discrimination*, which corresponds to the German words *keine* and *diskriminierung*, which do not form a contiguous sub-string in the German. Consider now a complete derivation, which derives the string *there is hierarchy of no discrimination*, and which includes the π-constituent *no discrimination* shown in the figure (i.e., where the treelet *discrimination* takes *no* as its only modifier). This derivation will violate the π-constituent constraint.[9]

5 Decoding

We now describe decoding algorithms for the syntactic models: we first describe inference rules that are used to combine pieces of structure, and then describe heuristic search algorithms that use these inference rules. Throughout this section, for brevity and simplicity, we describe algorithms that apply under the assumption that each s-phrase has a single associated treelet. The generalization to the case where an s-phrase may have multiple treelets is discussed in section 5.3.

5.1 Inference Rules

Parsing operations for the TAG grammars described in (Carreras et al., 2008) are based on the dynamic programming algorithms in (Eisner, 2000). A critical idea in dynamic programming algorithms such as these is to associate constituents in a chart with *spans* of the input sentence, and to introduce inference rules that combine constituents into larger pieces of structure. The crucial step in generalizing these algorithms to the non-projective case, and to translation, will be to make use of *bit-strings* that keep track of which words in the German have already been translated in a chart entry. To return to the example from the introduction, again assume that the selected s-phrases

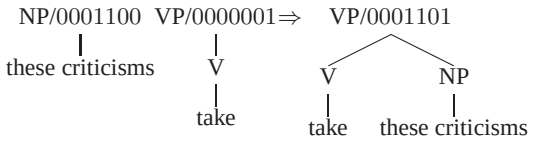
[9]Note, however, that the derivation step show in figure 4 will be considered in the search, because if *discrimination* takes additional modifiers, and thereby forms a π-constituent that dominates a contiguous sub-string in the German, then the resulting derivation will be valid.

0. Data structures: $\mathcal{Q}_i$ for $i = 1 \ldots n$ is a set of hypotheses for each length i, $\mathcal{S}$ is a set of chart entries
1. $\mathcal{S} \leftarrow \emptyset$
2. Initialize $\mathcal{Q}_1 \ldots \mathcal{Q}_n$ with basic chart entries derived from phrase entries
3. **For** $i = 1 \ldots n$
4. **For** any $A \in \text{BEAM}(\mathcal{Q}_i)$
5. **If** $\mathcal{S}$ contains a chart entry with the same signature as A, and which has a higher inside score,
6. **continue**
7. **Else**
8. Add A to $\mathcal{S}$
9. For any chart entry C that can be derived from A together with another chart entry $B \in \mathcal{S}$, add C to the set $\mathcal{Q}_j$ where $j = \text{length}(C)$
10. **Return** $\mathcal{Q}_n$, a set of items of length n

Figure 6: A beam search algorithm. A dynamic-programming *signature* consists of the regular dynamic-programming state for the parsing algorithm, together with the span (bit-string) associated with a constituent.

segment the German input into *[wir müssen auch] [diese kritik] [ernst] [nehmen]*, and the treelets are as shown in the introduction. Each of these treelets will form a basic entry in the chart, and will have an associated bit-string indicating which German words have been translated by that entry.

These basic chart entries can then be combined to form larger pieces of structure. For example, the following inferential step is possible:



We have shown the bit-string representation for each consituent: for example, the new constituent has the bit-string 0001101 representing the fact that the non-contiguous sub-strings *diese kritik* and *nehmen* have been translated at this point. Any two constituents can be combined, providing that the logical AND of their bit-strings is all 0's.

Inference steps such as that shown above will have an associated score corresponding to the TAG adjunction that is involved: in our models, both $score_{SYN}$ and $score_R$ will contribute to this score. In addition, we add state—specifically, word bigrams at the start and end of constituents—that allows trigram language model scores to be calculated as constituents are combined.

5.2 Approximate Search

There are 2^n possible bit-strings for a sentence of length n, hence the search space is of exponential size; approximate algorithms are therefore required in search for the highest scoring derivation. Figure 6 shows a beam search algorithm which makes use of the inference rules described in the

previous section. The algorithm stores sets $\mathcal{Q}_i$ for $i = 1 \ldots n$, where n is the source-language sentence length; each set $\mathcal{Q}_i$ stores hypotheses of length i (i.e., hypotheses with an associated bit-string with i ones). These sets are initialized with basic entries derived from s-phrases.

The function BEAM($\mathcal{Q}_i$) returns all items within $\mathcal{Q}_i$ that have a high enough score to fall within a beam (more details for BEAM are given below). At each iteration (step 4), each item in turn is taken from BEAM($\mathcal{Q}_i$) and added to a chart; the inference rules described in the previous section are used to derive new items which are added to the appropriate set $\mathcal{Q}_j$, where $j > i$.

We have found the definition of BEAM($\mathcal{Q}_i$) to be critical to the success of the method. As a first step, each item in $\mathcal{Q}_i$ receives a score that is a sum of an inside score (the cost of all derivation steps used to create the item) and a future score (an estimate of the cost to complete the translation). The future score is based on the source-language words that are still to be translated—this can be directly inferred from the item's bit-string—this is similar to the use of future scores in Pharoah (Koehn et al., 2003), and in fact we use Pharoah's future scores in our model. We then give the following definition, where N is a parameter (the beam size):

Definition 4 *(BEAM) Given $\mathcal{Q}_i$, define $\mathcal{Q}_{i,j}$ for $j = 1 \ldots n$ to be the subset of items in $\mathcal{Q}_i$ which have their j'th bit equal to one (i.e., have the j'th source language word translated). Define $\mathcal{Q}'_{i,j}$ to be the N highest scoring elements in $\mathcal{Q}_{i,j}$. Then BEAM($\mathcal{Q}_i$) $= \cup_{j=1}^{n} \mathcal{Q}'_{i,j}$.*

To motivate this definition, note that a naive method would simply define BEAM($\mathcal{Q}_i$) to be the N highest scoring elements of $\mathcal{Q}_i$. This definition, however, assumes that constituents which form translations of different parts of a sentence have scores that can be compared—an assumption that would be true if the future scores were highly accurate, but which quickly breaks down when future scores are inaccurate. In contrast, the definition above ensures that the top N analyses for each of the n source language words are stored at each stage, and hence that all parts of the source sentence are well represented. In experiments, the naive approach was essentially a failure, with parsing of some sentences either failing or being hopelessly inefficient, depending on the choice of N. In contrast, definition 4 gives good results.

System	BLEU score
Syntax-based	25.2
Syntax (no $Score_R$)	23.7 (-1.5)
Syntax (no π-c constraint)	24.4 (-0.8)

Table 1: Development set results showing the effect of removing $Score_R$ or the π-constituent constraint.

5.3 Allowing Multiple Treelets per s-Phrase

The decoding algorithms that we have described apply in the case where each s-phrase has a single treelet. The extension of these algorithms to the case where a phrase may have multiple treelets (e.g., see figure 2) is straightforward, but for brevity the details are omitted. The basic idea is to extend bit-string representations with a record of "pending" treelets which have not yet been included in a derivation. It is also possible to enforce the π-constituent constraint during decoding, as well as a constraint that ensures that reordering operations do not "break apart" English sub-strings within s-phrases that have multiple treelets (for example, for the s-phrase in figure 2, we ensure that *there is no* remains as a contiguous sequence of words in any translation using this s-phrase).

6 Experiments

We trained the syntax-based system on 751,088 German-English translations from the Europarl corpus (Koehn, 2005). A syntactic language model was also trained on the English sentences in the training data. We used Pharoah (Koehn et al., 2003) as a baseline system for comparison; the s-phrases used in our system include all phrases, with the same scores, as those used by Pharoah, allowing a direct comparison. For efficiency reasons we report results on sentences of length 30 words or less.[10] The syntax-based method gives a BLEU (Papineni et al., 2002) score of 25.04, a 0.46 BLEU point gain over Pharoah. This result was found to be significant ($p = 0.021$) under the paired bootstrap resampling method of Koehn (2004), and is close to significant ($p = 0.058$) under the sign test of Collins et al. (2005).

Table 1 shows results for the full syntax-based system, and also results for the system with the discriminative dependency scores (see section 4.1) and the π-contituent constraint removed from the system. In both cases we see a clear impact of these components of the model, with 1.5 and 0.8 BLEU point decrements respectively.

[10] Both Pharoah and our system have weights trained using MERT (Och, 2003) on sentences of length 30 words or less, to ensure that training and test conditions are matched.

R: in our eyes , the opportunity created by this directive of introducing longer buses on international routes is efficient .
S: the opportunity now presented by this directive is effective in our opinion , to use long buses on international routes .
P: the need for this directive now possibility of longer buses on international routes to is in our opinion , efficiently .
R: europe and asia must work together to intensify the battle against drug trafficking , money laundering , international crime , terrorism and the sexual exploitation of minors .
S: europe and asia must work together in order to strengthen the fight against drug trafficking , money laundering , against international crime , terrorism and the sexual exploitation of minors .
P: europe and asia must cooperate in the fight against drug trafficking , money laundering , against international crime , terrorism and the sexual exploitation of minors strengthened .
R: equally important for the future of europe - at biarritz and later at nice - will be the debate on the charter of fundamental rights .
S: it is equally important for the future of europe to speak on the charter of fundamental rights in biarritz , and then in nice .
P: just as important for the future of europe , it will be in biarritz and then in nice on the charter of fundamental rights to speak .
R: the convention was thus a muddled system , generating irresponsibility , and not particularly favourable to well-ordered democracy .
S: therefore , the convention has led to a system of a promoter of irresponsibility of the lack of clarity and hardly coincided with the rules of a proper democracy .
P: the convention therefore led to a system of full of lack of clarity and hardly a promoter of the irresponsibility of the rules of orderly was a democracy .

Figure 7: Examples where both annotators judged the syntactic system to give an improved translation when compared to the baseline system. 51 out of 200 translations fall into this category. These examples were chosen at random from these 51 examples. **R** is the human (reference) translation; **S** is the translation from the syntax-based system; **P** is the output from the baseline (phrase-based) system.

	Syntax	PB	=	Total
Syntax	51	3	7	61
PB	1	25	11	37
=	21	14	67	102
Total	73	42	85	200

Table 2: Human annotator judgements. Rows show results for annotator 1, and columns for annotator 2. *Syntax* and *PB* show the number of cases where an annotator respectively preferred/dispreferred the syntax-based system. $=$ gives counts of translations judged to be equal in quality.

In addition, we obtained human evaluations on 200 sentences chosen at random from the test data, using two annotators. For each example, the reference translation was presented to the annotator, followed by translations from the syntax-based and phrase-based systems (in a random order). For each example, each annotator could either decide that the two translations were of equal quality, or that one translation was better than the other. Table 2 shows results of this evaluation. Both annotators show a clear preference for the syntax-based system: for annotator 1, 73 translations are judged to be better for the syntax-based system, with 42 translations being worse; for annotator 2, 61 translations are improved with 37 being worse; both annotators' results are statistically significant with $p < 0.05$ under the sign test. Figure 7 shows some translation examples where the syntax-based system was judged to give an improvement.

7 Conclusions and Future Work

We have described a translation model that makes use of flexible parsing operations, critical ideas being the definition of s-phrases, Γ-dependencies, the π-constituent constraint, and an approximate search algorithm. A key area for future work will be further development of the discriminative dependency model (section 4.1). The model of $score_r(\gamma, \mathbf{f})$ that we have described in this paper is relatively simple; in general, however, there is the potential for $score_r$ to link target language dependencies to arbitrary properties of the source language string $\mathbf{f}$ (recall that γ contains a head and modifier spine in the target language, along with positions in the source-language string to which these spines are aligned). For example, we might introduce features that: a) condition dependencies created in the target language on dependency relations between their aligned words in the source language; b) condition target-language dependencies on whether they are aligned to words that are in the same clause or segment in the source language string; or, c) condition the grammatical roles of nouns in the target language on grammatical roles of aligned words in the source language. These features should improve translation quality by giving a tighter link between syntax in the source and target languages, and would be easily incorporated in the approach we have described.

Acknowledgments We would like to thank Ryan McDonald for conversations that were influential in this work, and Meg Aycinena Lippow and Ben Snyder for translation judgments. This work was supported under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022.

References

H. Alshawi. 1996. Head automata and bilingual tiling: Translation with minimal representations. In *Proceedings of ACL*, pages 167–176.

X. Carreras, M. Collins, and T. Koo. 2008. TAG, dynamic programming and the perceptron for efficient, feature-rich parsing. In *Proc. of CoNLL*.

E. Charniak, K. Knight, and K. Yamada. 2003. Syntax-based language models for machine translation. In *Proceedings of MT Summit IX*.

E. Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of ACL 2001*.

C. Cherry. 2008. Cohesive phrase-based decoding for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 72–80, Columbus, Ohio, June. Association for Computational Linguistics.

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

M. Collins, P. Koehn, and I. Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain, July. Association for Computational Linguistics.

J. Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In H. C. Bunt and A. Nijholt, editors, *New Developments in Natural Language Parsing*, pages 29–62. Kluwer Academic Publishers.

J. Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of ACL*.

A.K. Joshi and Y. Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and K. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 169–124. Springer.

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

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*.

D. Marcu, W. Wang, A. Echihabi, and K. Knight. 2006. Spmt: Statistical machine translation with syntactified target language phrases. In *Proceedings of EMNLP*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

D. Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of ACL*.

H. Mi, L. Huang, and Q. Liu. 2008. Forest-based translation. In *Proceedings of ACL-08: HLT*, pages 192–199. Association for Computational Linguistics.

R. Nesson, S.M. Shieber, and A. Rush. 2006. Induction of probabilistic synchronous tree-insertion grammars for machine translation. In *Proceedings of the 7th AMTA*.

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

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

C. Quirk, A. Menezes, and Colin Cherry. 2005. Dependency tree translation: Syntactically informed phrasal smt. In *Proceedings of ACL*.

O. Rambow, K. Vijay-Shanker, and D. Weir. 1995. D-tree grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 151–158, Cambridge, Massachusetts, USA, June. Association for Computational Linguistics.

L. Shen, J. Xu, and R. Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL*.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

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

H. Zhang and D. Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of ACL*, pages 473–482.

A. Zollmann and A. Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of NAACL 2006 Workshop on Statistical Machine Translation*.

Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models

Arne Mauser and **Saša Hasan** and **Hermann Ney**
Human Language Technology and Pattern Recognition Group
Chair of Computer Science 6, RWTH Aachen University, Germany
<surname>@cs.rwth-aachen.de

Abstract

In this work, we propose two extensions of standard word lexicons in statistical machine translation: A discriminative word lexicon that uses sentence-level source information to predict the target words and a trigger-based lexicon model that extends IBM model 1 with a second trigger, allowing for a more fine-grained lexical choice of target words. The models capture dependencies that go beyond the scope of conventional SMT models such as phrase- and language models. We show that the models improve translation quality by 1% in BLEU over a competitive baseline on a large-scale task.

1 Introduction

Lexical dependencies modeled in standard phrase-based SMT are rather local. Even though the decision about the best translation is made on sentence level, phrase models and word lexicons usually do not take context beyond the phrase boundaries into account. This is especially problematic since the average source phrase length used during decoding is small. When translating Chinese to English, e.g., it is typically close to only two words.

The target language model is the only model that uses lexical context across phrase boundaries. It is a very important feature in the log-linear setup of today's phrase-based decoders. However, its context is typically limited to three to six words and it is not informed about the source sentence. In the presented models, we explicitly take advantage of sentence-level dependencies including the source side and make non-local predictions for the target words. This is an important aspect when translating from languages like German and Chinese where long-distance dependencies are common. In Chinese, for example, tenses are often encoded by indicator words and particles whose position is relatively free in the sentence. In German, prefixes of verbs can be moved over long distances towards the end of the sentence.

In this work, we propose two models that can be categorized as extensions of standard word lexicons: A discriminative word lexicon that uses global, i.e. sentence-level source information to predict the target words using a statistical classifier and a trigger-based lexicon model that extends the well-known IBM model 1 (Brown et al., 1993) with a second trigger, allowing for a more fine-grained lexical choice of target words. The log-linear framework of the discriminative word lexicon offers a high degree of flexibility in the selection of features. Other sources of information such as syntax or morphology can be easily integrated.

The trigger-based lexicon model, or simply triplet model since it is based on word triplets, is not trained discriminatively but uses the classical maximum likelihood approach (MLE) instead. We train the triplets iteratively on a training corpus using the Expectation-Maximization (EM) algorithm. We will present how both models allow for a representation of topic-related sentence-level information which puts them close to word sense disambiguation (WSD) approaches. As will be shown later, the experiments indicate that these models help to ensure translation of content words that are often omitted by the baseline system. This is a common problem in Chinese-English translation. Furthermore, the models are often capable to produce a better lexical choice of content words.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 210–218,
Singapore, 6-7 August 2009.

The structure of the paper is as follows: In Section 2, we will address related work and briefly pin down how our models differentiate from previous work. Section 3 will describe the discriminative lexical selection model and the triplet model in more detail, explain the training procedures and show how the models are integrated into the decoder. The experimental setup and results will be given in Section 4. A more detailed discussion will be presented in Section 5. In the end, we conclude our findings and give an outlook for further research in Section 6.

2 Related Work

Several word lexicon models have emerged in the context of multilingual natural language processing. Some of them were used as a machine translation system or as a part of one such system. There are three major types of models: Heuristic models as in (Melamed, 2000), generative models as the IBM models (Brown et al., 1993) and discriminative models (Varea et al., 2001; Bangalore et al., 2006).

Similar to this work, the authors of (Varea et al., 2001) try to incorporate a maximum entropy lexicon model into an SMT system. They use the words and word classes from the local context as features and show improvements with n-best rescoring.

The models in this paper are also related to word sense disambiguation (WSD). For example, (Chan et al., 2007) trained a discriminative model for WSD using local but also across-sentence unigram collocations of words in order to refine phrase pair selection dynamically by incorporating scores from the WSD classifier. They showed improvements in translation quality in a hierarchical phrase-based translation system. Another WSD approach incorporating context-dependent phrasal translation lexicons is given in (Carpuat and Wu, 2007) and has been evaluated on several translation tasks. Our model differs from the latter in three ways. First, our approach models word selection of the target sentence based on global sentence-level features of the source sentence. Second, instead of disambiguating phrase senses as in (Carpuat and Wu, 2007), we model word selection independently of the phrases used in the MT models. Finally, the training is done in a different way as will be presented in Sections 3.1.1 and 3.2.1.

Recently, full translation models using discriminative training criteria emerged as well. They are designed to generate a translation for a given source sentence and not only score or disambiguate hypotheses given by a translation system. In (Ittycheriah and Roukos, 2007), the model can predict 1-to-many translations with gaps and uses words, morphologic and syntactic features from the local context.

The authors of (Venkatapathy and Bangalore, 2007) propose three different models. The first one is a global lexical selection model which includes all words of the source sentence as features, regardless of their position. Using these features, the system predicts the words that should be included in the target sentence. Sentence structure is then reconstructed using permutations of the generated bag of target words. We will also use this type of features in our model.

One of the simplest models in the context of lexical triggers is the IBM model 1 (Brown et al., 1993) which captures lexical dependencies between source and target words. It can be seen as a lexicon containing correspondents of translations of source and target words in a very broad sense since the pairs are trained on the full sentence level. The trigger-based lexicon model used in this work follows the training procedure introduced in (Hasan et al., 2008) and is integrated directly in the decoder instead of being applied in n-best list reranking. The model is very close to the IBM model 1 and can be seen as an extension of it by taking another word into the conditioning part, i.e. the triggering items. Thus, instead of $p(f|e)$, it models $p(f|e, e')$. Furthermore, since the second trigger can come from any part of the sentence, there is a link to long-range monolingual triggers as presented in (Tillmann and Ney, 1997) where a trigger language model was trained using the EM algorithm and helped to reduce perplexities and word error rates in a speech recognition experiment. In (Rosenfeld, 1996), another approach was chosen to model monolingual triggers using a maximum-entropy based framework. Again, this adapted LM could improve speech recognition performance significantly.

A comparison of a variant of the trigger-based lexicon model applied in decoding and n-best list reranking can be found in (Hasan and Ney, 2009). In order to reduce the number of overall triplets, the authors use the word alignments for fixing the

first trigger to the aligned target word. In general, this constraint performs slightly worse than the unconstrained variant used in this work, but allows for faster training and decoding.

3 Extended Lexicon Models

In this section, we present the extended lexicon models, how they are trained and integrated into the phrase-based decoder.

3.1 Discriminative Lexicon Model

Discriminative models have been shown to outperform generative models on many natural language processing tasks. For machine translation, however, the adaptation of these methods is difficult due to the large space of possible translations and the size of the training data that has to be used to achieve significant improvements.

In this section, we propose a discriminative word lexicon model that follows (Bangalore et al., 2007) and integrate it into the standard phrase-based machine translation approach.

The core of our model is a classifier that predicts target words, given the words of the source sentence. The structure of source as well as target sentence is neglected in this model. We do not make any assumptions about the location of the words in the sentence. This is useful in many cases, as words and morphology can depend on information given at other positions in the sentence. An example would be the character 了 in Chinese that indicates a completed or past action and does not need to appear close to the verb.

We model the probability of the set of target words in a sentence $\mathbf{e}$ given the set of source words $\mathbf{f}$. For each word in the target vocabulary, we can calculate a probability for being or not being included in the set. The probability of the whole set then is the product over the entire target vocabulary $\mathbf{V_E}$:

$$P(\mathbf{e}|\mathbf{f}) = \prod_{e \in \mathbf{e}} P(e^+|\mathbf{f}) \cdot \prod_{e \in \mathbf{V_E} \setminus \mathbf{e}} P(e^-|\mathbf{f}) \quad (1)$$

For notational simplicity, we use the event e^+ when the target word e is included in the target sentence and e^- if not. We model the individual factors $p(e|\mathbf{f})$ of the probability in Eq. 1 as a log-linear model using the source words from $\mathbf{f}$ as binary features

$$\phi(f, \mathbf{f}) = \begin{cases} 1 & \text{if } f \in \mathbf{f} \\ 0 & \text{else} \end{cases} \quad (2)$$

and feature weights $\lambda_{f,\cdot}$:

$$P(e^+|\mathbf{f}) = \frac{\exp\left(\sum_{f \in \mathbf{f}} \lambda_{f,e^+} \, \phi(f, \mathbf{f})\right)}{\sum_{e \in \{e^+, e^-\}} \exp\left(\sum_{f \in \mathbf{f}} \lambda_{f,e} \, \phi(f, \mathbf{f})\right)} \quad (3)$$

Subsequently, we will call this model discriminative word lexicon (DWL).

Modeling the lexicon on sets and not on sequences has two reasons. Phrase-based MT along with n-gram language models is strong at predicting sequences but only uses information from a local context. By using global features and predicting words in a non-local fashion, we can augment the strong local decisions from the phrase-based systems with sentence-level information.

For practical reasons, translating from a set to a set simplifies the parallelization of the training procedure. The classifiers for the target words can be trained separately as explained in the following section.

3.1.1 Training

Common classification tasks have a relatively small number of classes. In our case, the number of classes is the size of the target vocabulary. For large translation tasks, this is in the range of a hundred thousand classes. It is far from what conventional out-of-the-box classifiers can handle.

The discriminative word lexicon model has the convenient property that we can train a separate model for each target word making parallelization straightforward. Discussions about possible classifiers and the choice of regularization can be found in (Bangalore et al., 2007). We used the freely available MegaM Toolkit[1] for training, which implements the L-BFGS method (Byrd et al., 1995). Regularization is done using Gaussian priors. We performed 100 iterations of the training algorithm for each word in the target vocabulary. This results in a large number of classifiers to be trained. For the Arabic-English data (cf. Section 4), the training took an average of 38 seconds per word. No feature cutoff was used.

3.1.2 Decoding

In search, we compute the model probabilities as an additional model in the log-linear model combination of the phrase-based translation approach. To reduce the memory footprint and startup time of the decoding process, we reduced the number of

[1] http://www.cs.utah.edu/~hal/megam/

parameters by keeping only large values $\lambda_{f,e}$ since smaller values tend to have less effect on the overall probability. In experiments we determined that we could safely reduce the size of the final model by a factor of ten without losing predictive power. In search, we compute the model probabilities as an additional model in the log-linear combination. When scoring hypotheses from the phrase-based system, we see the translation hypothesis as the set of target words that are predicted. Words from the target vocabulary which are not included in the hypothesis are not part of the set. During the search process, however, we also have to score incomplete hypotheses where we do not know which words will not be included. This problem is circumvented by rewriting Eq. 1 as

$$P(\mathbf{e}|\mathbf{f}) = \prod_{e \in \mathbf{V_E}} P(e^-|\mathbf{f}) \cdot \prod_{e \in \mathbf{e}} \frac{P(e^+|\mathbf{f})}{P(e^-|\mathbf{f})}.$$

The first product is constant given a source sentence and therefore does not affect the search. Using the model assumption from Eq. 3, we can further simplify the computation and compute the model score entirely in log-space which is numerically stable even for large vocabularies. Experiments showed that using only the first factor of Eq. 1 is sufficient to obtain good results.

In comparison with the translation model from (Bangalore et al., 2007) where a threshold on the probability is used to determine which words are included in the target sentence, our approach relies on the phrase model to generate translation candidates. This has several advantages: The length of the translation is determined by the phrase model. Words occurring multiple times in the translation do not have to be explicitly modeled. In (Bangalore et al., 2007), repeated target words are treated as distinct classes.

The main advantage of the integration being done in a way as presented here is that the phrase model and the discriminative word lexicon model are complementary in the way they model the translation. While the phrase model is good in predicting translations in a local context, the discriminative word lexicon model is able to predict global aspects of the sentence like tense or vocabulary changes in questions. While the phrase model is closely tied to the structure of word and phrase alignments, the discriminative word lexicon model completely disregards the structure in source and target sentences.

3.2 Trigger-based Lexicon Model

The triplets of the trigger-based lexicon model, i.e. $p(e|f, f')$, are composed of two words in the source language triggering one target language word. We chose this inverse direction since it can be integrated directly into the decoder and, thus, does not rely on a two-pass approach using reranking, as it is the case for (Hasan et al., 2008). The triggers can originate from words of the whole source sentence, also crossing phrase boundaries of the conventional bilingual phrase pairs. The model is symmetric though, meaning that the order of the triggers is not relevant, i.e. $(f, f' \rightarrow e) = (f', f \rightarrow e)$. Nevertheless, the model is able to capture long-distance effects such as verb splits or adjustments to lexical choice of the target word given the topic-triggers of the source sentence. In training, we determine the probability of a target sentence e_1^I given the source sentence f_1^J within the model by

$$\begin{aligned} p(e_1^I|f_1^J) &= \prod_{i=1}^{I} p(e_i|f_1^J) \\ &= \prod_{i=1}^{I} \frac{2}{J(J+1)} \sum_{j=0}^{J} \sum_{j'=j+1}^{J} p(e_i|f_j, f_{j'}), \quad (4) \end{aligned}$$

where f_0 denotes the empty word and, thus, for $f_j = \varepsilon$, allows for modeling the conventional (inverse) IBM model 1 lexical probabilities as well. Since the second trigger $f_{j'}$ always starts right of the current first trigger, the model is symmetric and does not need to look at all trigger pairs. Eq. 4 is used in the iterative EM training on all sentence pairs of the training data which is described in more detail in the following.

3.2.1 Training

For training the trigger-based lexicon model, we apply the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). The goal is to maximize the log-likelihood F_{trip} of this model for a given bilingual training corpus $\{(f_1^{J_n}, e_1^{I_n})\}_1^N$ consisting of N sentence pairs:

$$F_{trip} := \sum_{n=1}^{N} \log p(e_1^{I_n}|f_1^{J_n}),$$

where I_n and J_n are the lengths of the n-th target and source sentence, respectively. An auxiliary function $Q(\mu; \bar{\mu})$ is defined based on F_{trip}

where $\bar{\mu}$ is the updated estimate within an iteration which is to be derived from the current estimate μ. Here, μ stands for the entire set of model parameters, i.e. the set of all $\{\alpha(e|f, f')\}$ with the constraint $\sum_e \alpha(e|f, f') = 1$. The accumulators $\alpha(\cdot)$ are therefore iteratively trained on the training data by using the current estimate, i.e. deriving the expected value (E-step), and maximizing their likelihood afterwards to reestimate the distribution. Thus, the perplexity of the training data is reduced in each iteration.

3.2.2 Decoding

In search, we can apply this model directly when scoring bilingual phrase pairs. Given a trained model for $p(e|f, f')$, we compute the feature score $h_{trip}(\cdot)$ of a phrase pair $(\tilde{e}, \tilde{f})$ as

$$h_{trip}(\tilde{e}, \tilde{f}, f_0^J) = - \sum_i \log \Big(\frac{2}{J \cdot (J+1)} \sum_j \sum_{j'>j} p(\tilde{e}_i | f_j, f_{j'}) \Big), \quad (5)$$

where i moves over all target words in the phrase $\tilde{e}$, the second sum selects all source sentence words f_0^J including the empty word, and $j' > j$ incorporates the rest of the source sentence right of the first trigger. We take negative log-probabilities and normalize to obtain the final score (representing costs) for the given phrase pair. Note that in search, we can only use this direction, $p(e|f, f')$, since the whole source sentence is available for triggering effects whereas not all target words have been generated so far, as it would be necessary for the standard direction, $p(f|e, e')$.

Due to the enormous number of triplets, we trained the model on a subset of the overall training data. The subcorpus, mainly consisting of newswire articles, contained 1.4M sentence pairs with 32.3M running words on the English side. We trained two versions of the triplet lexicon, one using 4 EM iterations and another one that was trained for 10 EM iterations. Due to trimming of triplets with small probabilities after each iteration, the version based on 10 iterations was slightly smaller, having 164 million triplets but also performed slightly worse. Thus, for the experiments, we used the version based on 4 iterations which contained 291 million triplets.

Note that decoding with this model can be quite efficient if caching is applied. Since the given source sentence does not change, we have to calculate $p(e|f, f')$ for each e only once and can retrieve the probabilities from the cache for consecutive scorings of the same target word e. This significantly speeds up the decoding process.

	train (C/E)	test08 (NW/WT)	
Sent. pairs	9.1M	480	490
Run. words	259M/300M	14.8K	12.3K
Vocabulary	357K/627K	3.6K	3.2K

Table 1: GALE Chinese-English corpus statistics including two test sets: newswire and web text.

	train C/E — A/E		nist08 C/A
Sent. pairs	7.3M	4.6M	1357
Words (M)	185/196	142/139	36K/46K
Vocab. (K)	163/265	351/361	6.4K/9.6K

Table 2: NIST Chinese-English and Arabic-English corpus statistics including the official 2008 test sets.

4 Experimental Evaluation

In this section we evaluate our lexicon models on the GALE Chinese-English task for newswire and web text translation and additionally on the official NIST 2008 task for both Chinese-English and Arabic-English. The baseline system was built using a state-of-the art phrase-based MT system (Zens and Ney, 2008). We use the standard set of models with phrase translation probabilities for source-to-target and target-to-source direction, smoothing with lexical weights, a word and phrase penalty, distance-based and lexicalized reordering and a 5-gram (GALE) or 6-gram (NIST) target language model.

We used training data provided by the Linguistic Data Consortium (LDC) consisting of 9.1M parallel Chinese-English sentence pairs of various domains for GALE (cf. Table 1) and smaller amounts of data for the NIST systems (cf. Table 2). The DWL and Triplet models were integrated into the decoder as presented in Section 3.

For the GALE development and test set, we separated the newswire and web text parts and did separate parameter tuning for each genre using the corresponding development set which consists of 485 sentences for newswire texts and 533 sentences of web text. The test set has 480 sentences for newswire and 490 sentences for web text. For NIST, we tuned on the official 2006 eval set and used the 2008 evaluation set as a blind test set.

GALE test08	NW		WT	
	BLEU [%]	TER [%]	BLEU [%]	TER [%]
Baseline	32.3	59.38	25.3	64.40
DWL	33.1	58.90	26.2	63.75
Triplet	32.9	58.59	26.2	64.20
DWL+Trip.	33.3	58.23	26.3	63.87

Table 3: Results on the GALE Chinese-English test set for the newswire and web text setting (case-insensitive evaluation).

NIST nist08	Chinese-Eng.		Arabic-Eng.	
	BLEU [%]	TER [%]	BLEU [%]	TER [%]
Baseline	26.8	65.11	42.0	50.55
DWL	27.6	63.56	42.4	50.01
Triplet	27.7	63.60	42.9	49.76
DWL+Trip.	27.9	63.56	43.0	49.15

Table 4: Results on the test sets for the NIST 2008 Chinese-English and Arabic-English task (case-insensitive evaluation).

4.1 Translation Results

The translation results on the two GALE test sets are shown in Table 3 for newswire and web text. Both the discriminative word lexicon and the triplet lexicon can individually improve the baseline by approximately +0.6–0.9% BLEU and -0.5–0.8% TER. For the combination of both lexicons on the newswire setting, we observe only a slight improvement on BLEU but also an additional boost in TER reduction, arriving at +1% BLEU and -1.2% TER. For web text, the findings are similar: The combination of the discriminative and trigger-based lexicons yields +1% BLEU and decreases TER by -0.5%.

We compared these results against an inverse IBM model 1 but the results were inconclusive which is consistent with the results presented in (Och et al., 2004) where no improvements were achieved using $p(e|f)$. In our case, inverse IBM1 improves results by 0.2–0.4% BLEU on the development set but does not show the same trend on the test sets. Furthermore, combining IBM1 with DWL or Triplets often even degraded the translation results, e.g. only 32.8% BLEU was achieved on newswire for a combination of the IBM1, DWL and Triplet model. In contrast, combinations of the DWL and Triplet model did not degrade performance and could benefit from each other.

In addition to the automatic scoring, we also did a randomized subjective evaluation where the hypotheses of the baseline was compared against the hypotheses generated using the discriminative word lexicon and triplet models. We evaluated 200 sentences from newswire and web text. In 80% of the evaluated sentences, the improved models were judged equal or better than the baseline.

We tested the presented lexicon models also on another large-scale system, i.e. NIST, for two language pairs, namely Chinese-English and Arabic-English. Interestingly, the results obtained for Arabic-English are similar to the findings for Chinese-English, as can be seen in Table 4. The overall improvements for this language pair are +1% BLEU and -1.4% TER. In contrast to the GALE Chinese-English task, the triplet lexicon model for the Arabic-English language pair performs slightly better than the discriminative word lexicon.

These results strengthen the claim that the presented models are capable of improving lexical choice of the MT system. In the next section, we discuss the observed effects and analyze our results in more detail.

5 Discussion

In terms of automatic evaluation measures, the results indicate that it is helpful to incorporate the extended lexicon models into the search process. In this section, we will analyze some more details of the models and take a look at the lexical choice they make and what differentiates them from the baseline models. In Table 5, we picked an example sentence from the GALE newswire test set and show the different hypotheses produced by our system. As can be seen, the baseline does not produce the present participle of the verb *restore* which makes the sentence somewhat hard to understand. Both the discriminative and the trigger-based lexicon approach are capable of generating this missing information, i.e. the correct use of *restoring*. Figure 1 gives an example how discontinuous triggers affect the word choice on the target side. Two cases are depicted where high probabilities of triplets including *emergency* and *restoring* on the target side influence the overall hypothesis selection. The non-local modeling advantages of the triplet model can be observed as well: The

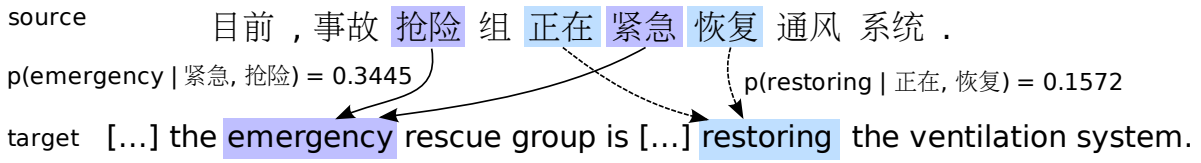


Figure 1: Triggering effect for the example sentence using the triplet lexicon model. The Chinese source sentence is shown in its segmented form. Two triplets are highlighted that have high probability and favor the target words *emergency* and *restoring*.

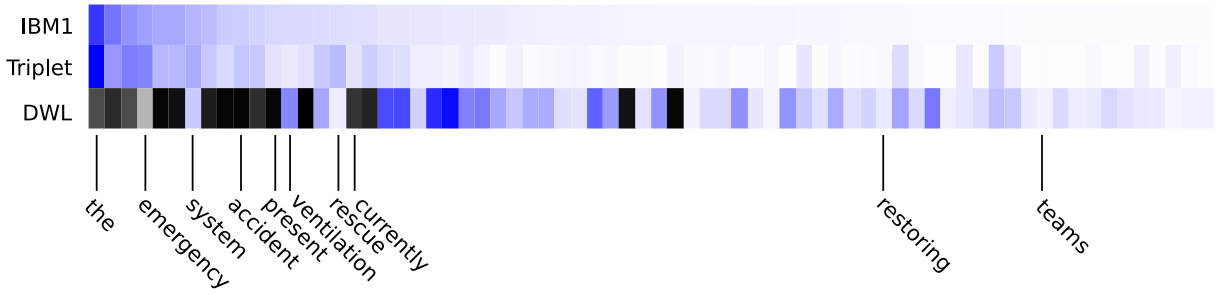


Figure 2: Ranking of words for the example sentence for IBM1, Triplet and DWL model. Ranks are sorted at IBM1, darker colors indicate higher probabilities within the model.

triggering events do not need to be located next to each other or within a given phrase pair. They move across the whole source sentence, thus allowing for capturing of long-range dependencies.

Table 6 shows the top ten content words that are predicted by the two models, discriminative word lexicon and triplet lexicon model. IBM model 1 ranks are indicated by subscripts in the column of the triplet model. Although the triplet model is similar to IBM1, we observe differences in the word lists. Comparing this to the visualization of the probability distribution for the example sentence, cf. Figure 2, we argue that, although the IBM1 and Triplet distributions look similar, the triplet model is sharper and favors words such as the ones in Table 6, resulting in different word choice in the translation process. In contrast, the DWL approach gives more distinct probabilities, selecting content words that are not chosen by the other models.

Table 7 shows an example from the web text test set. Here, the baseline hypothesis contains an incorrect word, *anna*, which might have been mistaken for the name *ying*. Interestingly, the hypotheses of the DWL lexicon and the combination of DWL and Triplet contain the correct content word *remarks*. The triplet model makes an error by selecting *music*, an artifact that might come from words that co-occur frequently with the corresponding Chinese verb *to listen*, i.e. 听 , in the data. Although the TER score of the baseline is better than the one for the alternative models for this particular example, we still think that the observed effects show how our models help producing different hypotheses that might lead to subjectively better translations.

An Arabic-English translation example is shown in Table 8. Here, the term *incidents of murder in apartments* was chosen over the baseline's *killings inside the flats*. Both translations are understandable and the difference in the wording is only based on synonyms. The translation using the discriminative and trigger-based lexicons better matches the reference translation and, thus, reflects a better lexical choice of the content words.

6 Conclusion

We have presented two lexicon models that use global source sentence context and are capable of predicting context-specific target words. The models have been directly integrated into the decoder and have shown to improve the translation quality of a state-of-the-art phrase-based machine translation system. The first model was a discriminative word lexicon that uses sentence-level features to predict if a word from the target vocabulary should be included in the translation or not. The second model was a trigger-based lexi-

Source	目前 , 事故 抢险 组 正在 紧急 恢复 通风 系统 .
Baseline	at present, the accident and rescue teams are currently emergency recovery ventilation systems.
DWL	at present, the emergency rescue teams are currently restoring the ventilation system.
Triplet	at present, the emergency rescue group is in the process of restoring the ventilation system.
DWL +Triplet	at present, the accident emergency rescue teams are currently restoring the ventilation system.
Reference	right now, the accident emergency rescue team is making emergency repair on the ventilation system.

Table 5: Translation example from the GALE newswire test set, comparing the baseline and the extended lexicon models given a reference translation. The Chinese source sentence is presented in its segmented form.

con that uses triplets to model long-range dependencies in the data. The source word triggers can move across the whole sentence and capture the topic of the sentence and incorporate more fine-grained lexical choice of the target words within the decoder.

Overall improvements are up to +1% in BLEU and -1.5% in TER on large-scale systems for Chinese-English and Arabic-English. Compared to the inverse IBM model 1 which did not yield consistent improvements, the presented models are valuable additional features in a phrase-based statistical machine translation system. We will test this setup for other language pairs and expect that languages like German where long-distance effects are common can benefit from these extended lexicon models.

In future work, we plan to extend the discriminative word lexicon model in two directions: extending context to the document level and feature engineering. For the trigger-based model, we plan to investigate more model variants. It might be interesting to look at cross-lingual trigger models such as $p(f|e, f')$ or constrained variants like $p(f|e, e')$ with $pos(e') < pos(e)$, i.e. the second trigger coming from the left context within a sentence which has already been generated. These

DWL		Triplet	
emergency	0.894	emergency_1	0.048
currently	0.330	system_2	0.032
current	0.175	rescue_8	0.027
emergencies	0.133	accident_3	0.022
present	0.133	ventilation_7	0.021
accident	0.119	work_{33}	0.021
recovery	0.053	present_5	0.011
group	0.046	currently_9	0.010
dealing	0.042	rush_{60}	0.010
ventilation	0.034	restoration_{31}	0.009

Table 6: The top 10 content words predicted by each model for the GALE newswire example sentence. Original ranks for the related IBM model 1 are given as subscripts for the triplet model.

Source	我 听 了 莹 的话 , 乐 得 哈哈大笑 .
Baseline	i have listened to anna, happy and laugh.
DWL	i have listened to the remarks, happy and laugh.
Triplet	i have listened to the music, a roar of laughter.
DWL +Triplet	i have listened to the remarks, happy and laugh.
Reference	hearing ying's remark, i laughed aloud happily.

Table 7: Translation example from the GALE web text test set. In this case, the baseline has a better TER but we can observe a corrected content word (*remark*) for the extended lexicon models. The Chinese source sentence is shown in its segmented form.

extensions could be integrated directly in search as well and would enable the system to combine both directions (standard and inverse) to some extent which was previously shown to help when applying the standard direction $p(f|e, e')$ as an additional reranking step, cf. (Hasan and Ney, 2009).

Acknowledgments

This material is partly based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023, and was partly realized as part of the Quaero Programme, funded by OSEO, French State agency for innovation.

The authors would like to thank Christian Buck

Source	و كانت بعض الصحف السعودية قد نشرت عددا من الحالات التي تعرضت ل السجن دون مبرر و كذلك بعض حوادث القتل داخل الشقق و غير ها .
Baseline	some saudi newspapers have published a number of cases that had been subjected to imprisonment without justification, as well as some killings inside the flats and others.
DWL +Triplet	some of the saudi newspapers have published a number of cases which were subjected to imprisonment without justification, as well as some incidents of murder in apartments and others.
Reference	some saudi newspapers have published a number of cases in which people were unjustifiably imprisoned, as well as some incidents of murder in apartments and elsewhere.

Table 8: Translation example from the NIST Arabic-English test set. The DWL and Triplet models improve lexical word choice by favoring *incidents of murder in apartments* instead of *killings inside the flats*. The Arabic source is shown in its segmented form.

and Juri Ganitkevitch for their help training the extended lexicon models.

References

S. Bangalore, P. Haffner, and S. Kanthak. 2006. Sequence classification for machine translation. In *Ninth International Conf. on Spoken Language Processing, Interspeech 2006 — ICSLP*, pages 1722–1725, Pitsburgh, PA, September.

S. Bangalore, P. Haffner, and S. Kanthak. 2007. Statistical machine translation through global lexical selection and sentence reconstruction. In *45th Annual Meeting of the Association of Computational Linguistics*, pages 152–159, Prague, Czech Republic, June.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–312, June.

R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

M. Carpuat and D. Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, Prague, Czech Republic, June.

Y. S. Chan, H. T. Ng, and D. Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *45th Annual Meeting of the Association of Computational Linguistics*, pages 33–40, Prague, Czech Republic, June.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–22.

S. Hasan and H. Ney. 2009. Comparison of extended lexicon models in search and rescoring for SMT. In *NAACL HLT 2009, Companion Volume: Short Papers*, pages 17–20, Boulder, Colorado, June.

S. Hasan, J. Ganitkevitch, H. Ney, and J. Andrés-Ferrer. 2008. Triplet lexicon models for statistical machine translation. In *EMNLP*, pages 372–381, Honolulu, Hawaii, October.

A. Ittycheriah and S. Roukos. 2007. Direct translation model 2. In *HLT-NAACL 2007: Main Conference*, pages 57–64, Rochester, New York, April.

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

F. J. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, and D. Radev. 2004. A smorgasbord of features for statistical machine translation. pages 161–168, Boston, MA, May.

R. Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10(3):187–228.

C. Tillmann and H. Ney. 1997. Word triggers and the EM algorithm. In *Proc. Special Interest Group Workshop on Computational Natural Language Learning (ACL)*, pages 117–124, Madrid, Spain, July.

I. García Varea, F. J. Och, H. Ney, and F. Casacuberta. 2001. Refined lexicon models for statistical machine translation using a maximum entropy approach. In *ACL '01: 39th Annual Meeting on Association for Computational Linguistics*, pages 204–211, Morristown, NJ, USA.

S. Venkatapathy and S. Bangalore. 2007. Three models for discriminative machine translation using global lexical selection and sentence reconstruction. In *SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 96–102, Rochester, New York, April.

R. Zens and H. Ney. 2008. Improvements in dynamic programming beam search for phrase-based statistical machine translation. In *International Workshop on Spoken Language Translation*, Honolulu, Hawaii, October.

Feature-Rich Translation by Quasi-Synchronous Lattice Parsing

Kevin Gimpel and **Noah A. Smith**
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{kgimpel,nasmith}@cs.cmu.edu

Abstract

We present a machine translation framework that can incorporate arbitrary features of both input and output sentences. The core of the approach is a novel decoder based on lattice parsing with quasi-synchronous grammar (Smith and Eisner, 2006), a syntactic formalism that does not require source and target trees to be isomorphic. Using generic approximate dynamic programming techniques, this decoder can handle "non-local" features. Similar approximate inference techniques support efficient parameter estimation with hidden variables. We use the decoder to conduct controlled experiments on a German-to-English translation task, to compare lexical phrase, syntax, and combined models, and to measure effects of various restrictions on non-isomorphism.

1 Introduction

We have seen rapid recent progress in machine translation through the use of rich features and the development of improved decoding algorithms, often based on grammatical formalisms.[1] If we view MT as a machine learning problem, features and formalisms imply structural independence assumptions, which are in turn exploited by efficient inference algorithms, including decoders (Koehn et al., 2003; Yamada and Knight, 2001). Hence a tension is visible in the many recent research efforts aiming to decode with "non-local" features (Chiang, 2007; Huang and Chiang, 2007).

Lopez (2009) recently argued for a separation between features/formalisms (and the independence assumptions they imply) from inference algorithms in MT; this separation is widely appreciated in machine learning. Here we take first steps toward such a "universal" decoder, making the following contributions:

Arbitrary feature model (§2): We define a single, direct log-linear translation model (Papineni et al., 1997; Och and Ney, 2002) that encodes most popular MT features and can be used to encode *any* features on source and target sentences, dependency trees, and alignments. The trees are optional and can be easily removed, allowing simulation of "string-to-tree," "tree-to-string," "tree-to-tree," and "phrase-based" models, among many others. We follow the widespread use of log-linear modeling for direct translation modeling; the novelty is in the use of richer feature sets than have been previously used in a single model.

Decoding as QG parsing (§3–4): We present a novel decoder based on lattice parsing with quasi-synchronous grammar (QG; Smith and Eisner, 2006).[2] Further, we exploit generic approximate inference techniques to incorporate arbitrary "non-local" features in the dynamic programming algorithm (Chiang, 2007; Gimpel and Smith, 2009).

Parameter estimation (§5): We exploit similar approximate inference methods in regularized pseudolikelihood estimation (Besag, 1975) with hidden variables to discriminatively and efficiently train our model. Because we start with *inference* (the key subroutine in training), many other learning algorithms are possible.

Experimental platform (§6): The flexibility of our model/decoder permits carefully controlled experiments. We compare lexical phrase and dependency syntax features, as well as a novel com-

[1] Informally, features are "parts" of a parallel sentence pair and/or their mutual derivation structure (trees, alignments, etc.). Features are often implied by a choice of formalism.

[2] To date, QG has been used for word alignment (Smith and Eisner, 2006), adaptation and projection in parsing (Smith and Eisner, 2009), and various monolingual recognition and scoring tasks (Wang et al., 2007; Das and Smith, 2009); this paper represents its first application to MT.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 219–228,
Singapore, 6-7 August 2009.

Σ, T	source and target language vocabularies, respectively
$\text{Trans} : \Sigma \cup \{\text{NULL}\} \rightarrow 2^{\text{T}}$	function mapping each source word to target words to which it may translate
$\boldsymbol{s} = \langle s_0, \ldots, s_n \rangle \in \Sigma^n$	source language sentence (s_0 is the NULL word)
$\boldsymbol{t} = \langle t_1, \ldots, t_m \rangle \in \text{T}^m$	target language sentence, translation of $\boldsymbol{s}$
$\tau_{\boldsymbol{s}} : \{1, \ldots, n\} \rightarrow \{0, \ldots, n\}$	dependency tree of $\boldsymbol{s}$, where $\tau_{\boldsymbol{s}}(i)$ is the index of the parent of s_i (0 is the root, \$)
$\tau_{\boldsymbol{t}} : \{1, \ldots, m\} \rightarrow \{0, \ldots, m\}$	dependency tree of $\boldsymbol{t}$, where $\tau_{\boldsymbol{t}}(i)$ is the index of the parent of t_i (0 is the root, \$)
$\boldsymbol{a} : \{1, \ldots, m\} \rightarrow 2^{\{1,\ldots,n\}}$	alignments from words in $\boldsymbol{t}$ to words in $\boldsymbol{s}$; $\emptyset$ denotes alignment to NULL
$\boldsymbol{\theta}$	parameters of the model
$\boldsymbol{g}_{trans}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{t})$	*lexical translation features (§2.1):*
$\boldsymbol{f}_{lex}(s, t)$	word-to-word translation features for translating s as t
$\boldsymbol{f}_{phr}(\boldsymbol{s}_i^j, \boldsymbol{t}_k^\ell)$	phrase-to-phrase translation features for translating $\boldsymbol{s}_i^j$ as $\boldsymbol{t}_k^\ell$
$\boldsymbol{g}_{lm}(\boldsymbol{t})$	*language model features (§2.2):*
$\boldsymbol{f}_N(\boldsymbol{t}_{j-N+1}^j)$	N-gram probabilities
$\boldsymbol{g}_{syn}(\boldsymbol{t}, \tau_{\boldsymbol{t}})$	*target syntactic features (§2.3):*
$\boldsymbol{f}_{att}(t, j, t', k)$	syntactic features for attaching target word t' at position k to target word t at position j
$\boldsymbol{f}_{val}(t, j, I)$	syntactic valence features with word t at position j having children $I \subseteq \{1, \ldots, m\}$
$\boldsymbol{g}_{reor}(\boldsymbol{s}, \tau_{\boldsymbol{s}}, \boldsymbol{a}, \boldsymbol{t}, \tau_{\boldsymbol{t}})$	*reordering features (§2.4):*
$\boldsymbol{f}_{dist}(i, j)$	distortion features for a source word at position i aligned to a target word at position j
$\boldsymbol{g}_{tree^2}(\tau_{\boldsymbol{s}}, \boldsymbol{a}, \tau_{\boldsymbol{t}})$	*tree-to-tree syntactic features (§3):*
$\boldsymbol{f}_{qg}(i, i', j, k)$	configuration features for source pair $s_i/s_{i'}$ being aligned to target pair t_j/t_k
$\boldsymbol{g}_{cov}(\boldsymbol{a})$	*coverage features (§4.2)*
$\boldsymbol{f}_{\boldsymbol{s}\,cov}(\boldsymbol{a})$, $\boldsymbol{f}_{z\text{th}}(\boldsymbol{a})$, $\boldsymbol{f}_{\boldsymbol{s}\,unc}(\boldsymbol{a})$	counters for "covering" each $\boldsymbol{s}$ word each time, the zth time, and leaving it "uncovered"

Table 1: Key notation. Feature factorings are elaborated in Tab. 2.

bination of the two. We quantify the effects of our approximate inference. We explore the effects of various ways of restricting syntactic non-isomorphism between source and target trees through the QG. We do not report state-of-the-art performance, but these experiments reveal interesting trends that will inform continued research.

2 Model

(Table 1 explains notation.) Given a sentence $\boldsymbol{s}$ and its parse tree $\tau_{\boldsymbol{s}}$, we formulate the translation problem as finding the target sentence $\boldsymbol{t}^*$ (along with its parse tree $\tau_{\boldsymbol{t}}^*$ and alignment $\boldsymbol{a}^*$ to the source tree) such that[3]

$$\langle \boldsymbol{t}^*, \tau_{\boldsymbol{t}}^*, \boldsymbol{a}^* \rangle = \underset{\langle \boldsymbol{t}, \tau_{\boldsymbol{t}}, \boldsymbol{a} \rangle}{\text{argmax}}\ p(\boldsymbol{t}, \tau_{\boldsymbol{t}}, \boldsymbol{a} \mid \boldsymbol{s}, \tau_{\boldsymbol{s}}) \quad (1)$$

In order to include overlapping features and permit hidden variables during training, we use a single globally-normalized conditional log-linear model. That is, $p(\boldsymbol{t}, \tau_{\boldsymbol{t}}, \boldsymbol{a} \mid \boldsymbol{s}, \tau_{\boldsymbol{s}}) =$

$$\frac{\exp\{\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{s}, \tau_{\boldsymbol{s}}, \boldsymbol{a}, \boldsymbol{t}, \tau_{\boldsymbol{t}})\}}{\sum_{\boldsymbol{a}', \boldsymbol{t}', \tau_{\boldsymbol{t}}'} \exp\{\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{s}, \tau_{\boldsymbol{s}}, \boldsymbol{a}', \boldsymbol{t}', \tau_{\boldsymbol{t}}')\}} \quad (2)$$

where the $\boldsymbol{g}$ are arbitrary feature functions and the $\boldsymbol{\theta}$ are feature weights. If one or both parse trees or the word alignments are unavailable, they can be ignored or marginalized out as hidden variables.

In a log-linear model over structured objects, the choice of feature functions $\boldsymbol{g}$ has a huge effect on the feasibility of inference, including decoding. Typically these feature functions are chosen to *factor* into local parts of the overall structure. We next define some key features used in current MT systems, explaining how they factor. We will use subscripts on $\boldsymbol{g}$ to denote different groups of features, which may depend on subsets of the structures $\boldsymbol{t}$, $\tau_{\boldsymbol{t}}$, $\boldsymbol{a}$, $\boldsymbol{s}$, and $\tau_{\boldsymbol{s}}$. When these features factor into parts, we will use $\boldsymbol{f}$ to denote the factored vectors, so that if $\boldsymbol{x}$ is an object that breaks into parts $\{x_i\}_i$, then $\boldsymbol{g}(\boldsymbol{x}) = \sum_i \boldsymbol{f}(x_i)$.[4]

2.1 Lexical Translations

Classical lexical translation features depend on $\boldsymbol{s}$ and $\boldsymbol{t}$ and the alignment $\boldsymbol{a}$ between them. The simplest are word-to-word features, estimated as the conditional probabilities $p(t \mid s)$ and $p(s \mid t)$ for $s \in \Sigma$ and $t \in \text{T}$. Phrase-to-phrase features generalize these, estimated as $p(\boldsymbol{t}' \mid \boldsymbol{s}')$ and $p(\boldsymbol{s}' \mid \boldsymbol{t}')$ where $\boldsymbol{s}'$ (respectively, $\boldsymbol{t}'$) is a substring of $\boldsymbol{s}$ ($\boldsymbol{t}$).

A major difference between the phrase features used in this work and those used elsewhere is that we do not assume that phrases segment into disjoint parts of the source and target sentences

[3] We assume in this work that $\boldsymbol{s}$ is parsed. In principle, we might include source-side parsing as part of decoding.

[4] There are two conventional definitions of feature functions. One is to let the range of these functions be *conditional probability* estimates (Och and Ney, 2002). These estimates are usually heuristic and inconsistent (Koehn et al., 2003). An alternative is to instantiate features for different structural patterns (Liang et al., 2006; Blunsom et al., 2008). This offers more expressive power but may require much more training data to avoid overfitting. For this reason, and to keep training fast, we opt for the former convention, though our decoder can handle both, and the factorings we describe are agnostic about this choice.

(Koehn et al., 2003); they can overlap.[5] Additionally, since phrase features can be any function of words and alignments, we permit features that consider phrase pairs in which a target word outside the target phrase aligns to a source word inside the source phrase, as well as phrase pairs with gaps (Chiang, 2005; Ittycheriah and Roukos, 2007).

Lexical translation features factor as in Eq. 3 (Tab. 2). We score all phrase pairs in a sentence pair that pair a target phrase with the smallest source phrase that contains all of the alignments in the target phrase; if $\bigcup_{k:i\leq k\leq j} \boldsymbol{a}(k) = \emptyset$, no phrase feature fires for $\boldsymbol{t}_i^j$.

2.2 N-gram Language Model

N-gram language models have become standard in machine translation systems. For bigrams and trigrams (used in this paper), the factoring is in Eq. 4 (Tab. 2).

2.3 Target Syntax

There have been many features proposed that consider source- and target-language syntax during translation. Syntax-based MT systems often use features on grammar rules, frequently maximum likelihood estimates of conditional probabilities in a probabilistic grammar, but other syntactic features are possible. For example, Quirk et al. (2005) use features involving phrases and source-side dependency trees and Mi et al. (2008) use features from a forest of parses of the source sentence. There is also substantial work in the use of target-side syntax (Galley et al., 2006; Marcu et al., 2006; Shen et al., 2008). In addition, researchers have recently added syntactic features to phrase-based and hierarchical phrase-based models (Gimpel and Smith, 2008; Haque et al., 2009; Chiang et al., 2008).

In this work, we focus on syntactic features of target-side dependency trees, $\tau_{\boldsymbol{t}}$, along with the words $\boldsymbol{t}$. These include attachment features that relate a word to its syntactic parent, and valence features. They factor as in Eq. 5 (Tab. 2). Features that consider only target-side syntax and words without considering $\boldsymbol{s}$ can be seen as "syntactic language model" features (Shen et al., 2008).

[5] Segmentation might be modeled as a hidden variable in future work.

$$\boldsymbol{g}_{trans}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{t}) = \sum_{j=1}^{m}\sum_{i\in\boldsymbol{a}(j)} \boldsymbol{f}_{lex}(s_i,t_j) + \sum_{i,j:1\leq i<j\leq m} \boldsymbol{f}_{phr}(\boldsymbol{s}_{first(i,j)}^{last(i,j)},\boldsymbol{t}_i^j) \quad (3)$$

$$\boldsymbol{g}_{lm}(\boldsymbol{t}) = \sum_{N\in\{2,3\}}\sum_{j=1}^{m+1} \boldsymbol{f}_N(\boldsymbol{t}_{j-N+1}^j) \quad (4)$$

$$\boldsymbol{g}_{syn}(\boldsymbol{t},\tau_{\boldsymbol{t}}) = \sum_{j=1}^{m} \boldsymbol{f}_{att}(t_j,j,t_{\tau_{\boldsymbol{t}}(j)},\tau_{\boldsymbol{t}}(j)) + \boldsymbol{f}_{val}(t_j,j,\tau_{\boldsymbol{t}}^{-1}(j)) \quad (5)$$

$$\boldsymbol{g}_{reor}(\boldsymbol{s},\tau_{\boldsymbol{s}},\boldsymbol{a},\boldsymbol{t},\tau_{\boldsymbol{t}}) = \sum_{j=1}^{m}\sum_{i\in\boldsymbol{a}(j)} \boldsymbol{f}_{dist}(i,j) \quad (6)$$

$$\boldsymbol{g}_{tree^2}(\tau_{\boldsymbol{s}},\boldsymbol{a},\tau_{\boldsymbol{t}}) = \sum_{j=1}^{m} \boldsymbol{f}_{qg}(\boldsymbol{a}(j),\boldsymbol{a}(\tau_{\boldsymbol{t}}(j)),j,\tau_{\boldsymbol{t}}(j)) \quad (7)$$

Table 2: Factoring of global feature collections $\boldsymbol{g}$ into $\boldsymbol{f}$. $\boldsymbol{x}_i^j$ denotes $\langle x_i,\ldots x_j\rangle$ in sequence $\boldsymbol{x} = \langle x_1,\ldots\rangle$. $first(i,j) = \min_{k:i\leq k\leq j}(\min(\boldsymbol{a}(k)))$ and $last(i,j) = \max_{k:i\leq k\leq j}(\max(\boldsymbol{a}(k)))$.

2.4 Reordering

Reordering features take many forms in MT. In phrase-based systems, reordering is accomplished both within phrase pairs (local reordering) as well as through distance-based distortion models (Koehn et al., 2003) and lexicalized reordering models (Koehn et al., 2007). In syntax-based systems, reordering is typically parameterized by grammar rules. For generality we permit these features to "see" all structures and denote them $\boldsymbol{g}_{reor}(\boldsymbol{s},\tau_{\boldsymbol{s}},\boldsymbol{a},\boldsymbol{t},\tau_{\boldsymbol{t}})$. Eq. 6 (Tab. 2) shows a factoring of reordering features based on absolute positions of aligned words.

We turn next to the "backbone" model for our decoder; the formalism and the properties of its decoding algorithm will inspire two additional sets of features.

3 Quasi-Synchronous Grammars

A *quasi-synchronous dependency grammar* (QDG; Smith and Eisner, 2006) specifies a conditional model $p(\boldsymbol{t},\tau_{\boldsymbol{t}},\boldsymbol{a} \mid \boldsymbol{s},\tau_{\boldsymbol{s}})$. Given a source sentence $\boldsymbol{s}$ and its parse $\tau_{\boldsymbol{s}}$, a QDG induces a probabilistic monolingual dependency grammar over sentences "inspired" by the source sentence and tree. We denote this grammar by $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$; its (weighted) language is the set of translations of $\boldsymbol{s}$. Each word generated by $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$ is annotated with a "sense," which consists of zero or more words from $\boldsymbol{s}$. The senses imply an alignment ($\boldsymbol{a}$) between words in $\boldsymbol{t}$ and words in $\boldsymbol{s}$, or equivalently, between nodes in $\tau_{\boldsymbol{t}}$ and nodes in $\tau_{\boldsymbol{s}}$. In principle, any portion of $\tau_{\boldsymbol{t}}$ may align to any portion of $\tau_{\boldsymbol{s}}$, but in practice we often make restrictions on the alignments to simplify computation. Smith and Eisner, for example, restricted $|\boldsymbol{a}(j)|$ for all words

t_j to be at most one, so that each target word aligned to at most one source word, which we also do here.[6]

Which translations are possible depends heavily on the *configurations* that the QDG permits. Formally, for a parent-child pair $\langle t_{\tau_{\boldsymbol{t}}(j)}, t_j \rangle$ in $\tau_{\boldsymbol{t}}$, we consider the relationship between $\boldsymbol{a}(\tau_{\boldsymbol{t}}(j))$ and $\boldsymbol{a}(j)$, the source-side words to which $t_{\tau_{\boldsymbol{t}}(j)}$ and t_j align. If, for example, we require that, for all j, $\boldsymbol{a}(\tau_{\boldsymbol{t}}(j)) = \tau_{\boldsymbol{s}}(\boldsymbol{a}(j))$ or $\boldsymbol{a}(j) = 0$, and that the root of $\tau_{\boldsymbol{t}}$ must align to the root of $\tau_{\boldsymbol{s}}$ or to NULL, then strict isomorphism must hold between $\tau_{\boldsymbol{s}}$ and $\tau_{\boldsymbol{t}}$, and we have implemented a synchronous CF dependency grammar (Alshawi et al., 2000; Ding and Palmer, 2005). Smith and Eisner (2006) grouped all possible configurations into eight classes and explored the effects of permitting different sets of classes in word alignment. ("$\boldsymbol{a}(\tau_{\boldsymbol{t}}(j)) = \tau_{\boldsymbol{s}}(\boldsymbol{a}(j))$" corresponds to their "parent-child" configuration; see Fig. 3 in Smith and Eisner (2006) for illustrations of the rest.) More generally, we can define *features* on tree pairs that factor into these local configurations, as shown in Eq. 7 (Tab. 2).

Note that the QDG instantiates the model in Eq. 2. Of the features discussed in §2, $\boldsymbol{f}_{lex}$, $\boldsymbol{f}_{att}$, $\boldsymbol{f}_{val}$, and $\boldsymbol{f}_{dist}$ can be easily incorporated into the QDG as described while respecting the independence assumptions implied by the configuration features. The others ($\boldsymbol{f}_{phr}$, $\boldsymbol{f}_2$, and $\boldsymbol{f}_3$) are *nonlocal*, or involve parts of the structure that, from the QDG's perspective, are conditionally independent given intervening material. Note that "nonlocality" is relative to a choice of formalism; in §2 we did not commit to any formalism, so it is only now that we can describe phrase and N-gram features as non-local. Non-local features will present a challenge for decoding and training (§4.3).

4 Decoding

Given a sentence $\boldsymbol{s}$ and its parse $\tau_{\boldsymbol{s}}$, at decoding time we seek the target sentence $\boldsymbol{t}^*$, the target tree $\tau_{\boldsymbol{t}}^*$, and the alignments $\boldsymbol{a}^*$ that are most probable, as defined in Eq. 1.[7] (In §5 we will consider k-best and all-translations variations on this problem.) As usual, the normalization constant is not required for decoding; it suffices to solve:

$$\langle \boldsymbol{t}^*, \tau_{\boldsymbol{t}}^*, \boldsymbol{a}^* \rangle = \operatorname*{argmax}_{\langle \boldsymbol{t}, \tau_{\boldsymbol{t}}, \boldsymbol{a} \rangle} \boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{s}, \tau_{\boldsymbol{s}}, \boldsymbol{a}, \boldsymbol{t}, \tau_{\boldsymbol{t}}) \quad (8)$$

For a QDG model, the decoding problem has not been addressed before. It equates to finding the most probable derivation under the $\boldsymbol{s}/\tau_{\boldsymbol{s}}$-specific grammar $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$. We solve this by lattice parsing, assuming that an upper bound on m (the length of $\boldsymbol{t}$) is known. The advantage offered by this approach (like most other grammar-based translation approaches) is that decoding becomes *dynamic programming* (DP), a technique that is both widely understood in NLP and for which practical, efficient, generic techniques exist. A major advantage of DP is that, with small modifications, *summing* over structures is also possible with "inside" DP algorithms. We will exploit this in training (§5). Efficient summing opens up many possibilities for training $\boldsymbol{\theta}$, such as likelihood and pseudo-likelihood, and provides principled ways to handle hidden variables during learning.

4.1 Translation as Monolingual Parsing

We decode by performing lattice parsing on a lattice encoding the set of possible translations. The lattice is a weighted "sausage" lattice that permits sentences up to some maximum length ℓ; ℓ is derived from the source sentence length. Let the states be numbered 0 to ℓ; states from $\lfloor \rho\ell \rfloor$ to ℓ are final states (for some $\rho \in (0, 1)$). For every position between consecutive states $j-1$ and j ($0 < j \leq \ell$), and for every word s_i in $\boldsymbol{s}$, and for every word $t \in \mathrm{Trans}(s_i)$, we instantiate an arc annotated with t and i. The weight of such an arc is $\exp\{\boldsymbol{\theta}^\top \boldsymbol{f}\}$, where $\boldsymbol{f}$ is the sum of feature functions that fire when s_i translates as t in target position j (e.g., $\boldsymbol{f}_{lex}(s_i, t)$ and $\boldsymbol{f}_{dist}(i, j)$).

Given the lattice and $G_{\boldsymbol{s},\tau_{\boldsymbol{s}}}$, lattice parsing is a straightforward generalization of standard context-free dependency parsing DP algorithms (Eisner, 1997). This decoder accounts for $\boldsymbol{f}_{lex}$, $\boldsymbol{f}_{att}$, $\boldsymbol{f}_{val}$, $\boldsymbol{f}_{dist}$, and $\boldsymbol{f}_{qg}$ as local features.

Figure 1 gives an example, showing a German sentence and dependency tree from an automatic parser, an English reference, and a lattice representing possible translations. In each bundle, the arcs are listed in decreasing order according to weight and for clarity only the first five are shown. The output of the decoder consists of lattice arcs

[6] I.e., from here on, $\boldsymbol{a} : \{1, \ldots, m\} \to \{0, \ldots, n\}$ where 0 denotes alignment to NULL.

[7] Arguably, we seek $\operatorname{argmax}_{\boldsymbol{t}} p(\boldsymbol{t} \mid \boldsymbol{s})$, marginalizing out everything else. Approximate solutions have been proposed for that problem in several settings (Blunsom and Osborne, 2008; Sun and Tsujii, 2009); we leave their combination with our approach to future work.

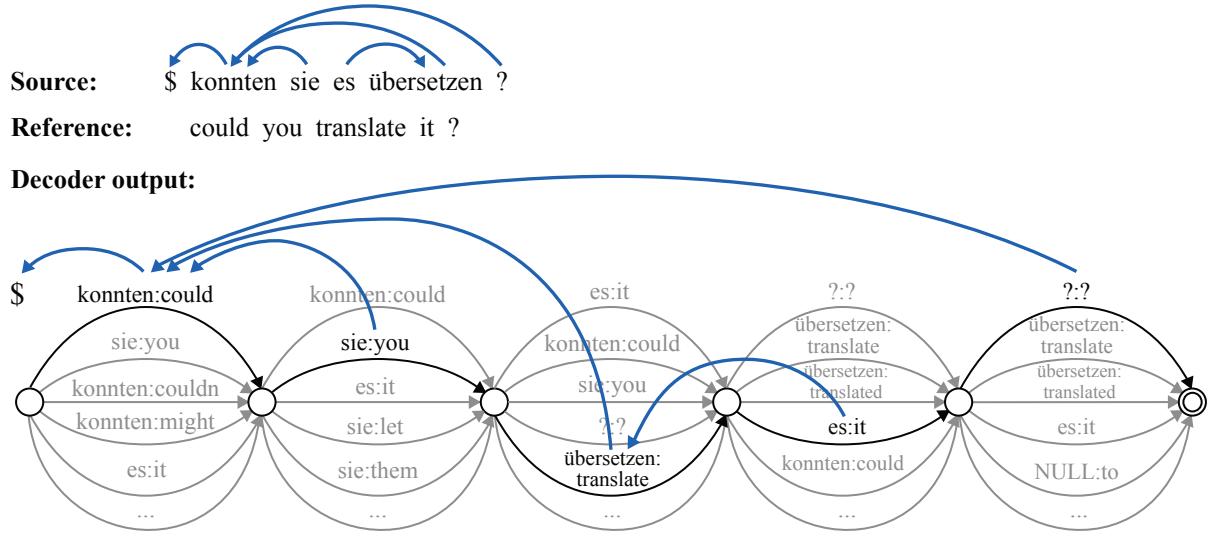


Figure 1: Decoding as lattice parsing, with the highest-scoring translation denoted by black lattice arcs (others are grayed out) and thicker blue arcs forming a dependency tree over them.

selected at each position and a dependency tree over them.

4.2 Source-Side Coverage Features

Most MT decoders enforce a notion of "coverage" of the source sentence during translation: all parts of $\boldsymbol{s}$ should be aligned to some part of $\boldsymbol{t}$ (alignment to NULL incurs an explicit cost). Phrase-based systems such as Moses (Koehn et al., 2007) explicitly search for the highest-scoring string in which all source words are translated. Systems based on synchronous grammars proceed by parsing the source sentence with the synchronous grammar, ensuring that every phrase and word has an analogue in $\tau_{\boldsymbol{t}}$ (or a deliberate choice is made by the decoder to translate it to NULL). In such systems, we do not need to use features to implement source-side coverage, as it is assumed as a hard constraint always respected by the decoder.

Our QDG decoder has no way to enforce coverage; it does not track any kind of state in $\tau_{\boldsymbol{s}}$ apart from a single recently aligned word. This is a problem with other direct translation models, such as IBM model 1 used as a direct model rather than a channel model (Brown et al., 1993). This sacrifice is the result of our choice to use a conditional model (§2).

The solution is to introduce a set of *coverage* features $\boldsymbol{g}_{cov}(\boldsymbol{a})$. Here, these include:

- A counter for the number of times each source word is covered: $\boldsymbol{f}_{\boldsymbol{s}cov}(\boldsymbol{a}) = \sum_{i=1}^{n} |\boldsymbol{a}^{-1}(i)|$.
- Features that fire once when a source word is covered the zth time ($z \in \{2, 3, 4\}$) and fire again all subsequent times it is covered; these are denoted $\boldsymbol{f}_{2\text{nd}}$, $\boldsymbol{f}_{3\text{rd}}$, and $\boldsymbol{f}_{4\text{th}}$.
- A counter of uncovered source words: $\boldsymbol{f}_{\boldsymbol{s}unc}(\boldsymbol{a}) = \sum_{i=1}^{n} \delta(|\boldsymbol{a}^{-1}(i)|, 0)$.

Of these, only $\boldsymbol{f}_{\boldsymbol{s}cov}$ is local.

4.3 Non-Local Features

The lattice QDG parsing decoder incorporates many of the features we have discussed, but not all of them. Phrase lexicon features $\boldsymbol{f}_{phr}$, language model features $\boldsymbol{f}_N$ for $N > 1$, and most coverage features are non-local with respect to our QDG. Recently Chiang (2007) introduced "cube pruning" as an approximate decoding method that extends a DP decoder with the ability to incorporate features that break the Markovian independence assumptions DP exploits. Techniques like cube pruning can be used to include the non-local features in our decoder.[8]

5 Training

Training requires us to learn values for the parameters $\boldsymbol{\theta}$ in Eq. 2. Given T training examples of the form $\langle \boldsymbol{t}^{(i)}, \tau_{\boldsymbol{t}}^{(i)}, \boldsymbol{s}^{(i)}, \tau_{\boldsymbol{s}}^{(i)} \rangle$, for $i = 1, ..., T$, maximum likelihood estimation for this model consists of solving Eq. 9 (Tab. 3).[9] Note that the

[8] A full discussion is omitted for space, but in fact we use "cube decoding," a slightly less approximate, slightly more expensive method that is more closely related to the approximate inference methods we use for training, discussed in §5.

[9] In practice, we regularize by including a term $-c\|\boldsymbol{\theta}\|_2^2$.

$$\mathrm{LL}(\boldsymbol{\theta}) = \sum_{i=1}^{T} \log p(\boldsymbol{t}^{(i)}, \tau_{\boldsymbol{t}}^{(i)} \mid \boldsymbol{s}^{(i)}, \tau_{\boldsymbol{s}}^{(i)}) = \sum_{i=1}^{T} \log \frac{\sum_{\boldsymbol{a}} \exp\{\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{s}^{(i)}, \tau_{\boldsymbol{s}}^{(i)}, \boldsymbol{a}, \boldsymbol{t}^{(i)}, \tau_{\boldsymbol{t}}^{(i)})\}}{\sum_{\boldsymbol{t}, \tau_{\boldsymbol{t}}, \boldsymbol{a}} \exp\{\boldsymbol{\theta}^\top \boldsymbol{g}(\boldsymbol{s}^{(i)}, \tau_{\boldsymbol{s}}^{(i)}, \boldsymbol{a}, \boldsymbol{t}, \tau_{\boldsymbol{t}})\}} = \sum_{i=1}^{T} \log \frac{\text{“numerator”}}{\text{“denominator”}} \quad (9)$$

$$\mathrm{PL}(\boldsymbol{\theta}) = \sum_{i=1}^{T} \log\Big(\sum_{\boldsymbol{a}} p(\boldsymbol{t}^{(i)}, \boldsymbol{a} \mid \tau_{\boldsymbol{t}}^{(i)}, \boldsymbol{s}^{(i)}, \tau_{\boldsymbol{s}}^{(i)})\Big) + \sum_{i=1}^{T} \log\Big(\sum_{\boldsymbol{a}} p(\tau_{\boldsymbol{t}}^{(i)}, \boldsymbol{a} \mid \boldsymbol{t}^{(i)}, \boldsymbol{s}^{(i)}, \tau_{\boldsymbol{s}}^{(i)})\Big) \quad (10)$$

$$\begin{array}{l}\text{“denominator” of}\\ \text{term 1 in Eq. 10}\end{array} = \sum_{i=0}^{n} \sum_{t' \in \mathrm{Trans}(s_i)} S(\tau_{\boldsymbol{t}}^{-1}(0), i, t') \times \exp\Big\{\boldsymbol{\theta}^\top \big(\boldsymbol{f}_{lex}(s_i, t') + \boldsymbol{f}_{att}(\$, 0, t', k) + \boldsymbol{f}_{qg}(0, i, 0, k)\big)\Big\} \quad (11)$$

$$S(j, i, t) = \prod_{k \in \tau_{\boldsymbol{t}}^{-1}(j)} \sum_{i'=0}^{n} \sum_{t' \in \mathrm{Trans}(s_{i'})} S(k, i', t') \times \exp\left\{\boldsymbol{\theta}^\top \begin{pmatrix} \boldsymbol{f}_{lex}(s_{i'}, t') + \boldsymbol{f}_{att}(t, j, t', k) + \\ \boldsymbol{f}_{val}(t, j, \tau_{\boldsymbol{t}}^{-1}(j)) + \boldsymbol{f}_{qg}(i, i', j, k) \end{pmatrix}\right\} \quad (12)$$

$$S(j, i, t) = \exp\Big\{\boldsymbol{\theta}^\top \big(\boldsymbol{f}_{val}(t, j, \tau_{\boldsymbol{t}}^{-1}(j))\big)\Big\} \quad \text{if } \tau_{\boldsymbol{t}}^{-1}(j) = \emptyset \quad (13)$$

Table 3: Eq. 9: Log-likelihood. Eq. 10: Pseudolikelihood. In both cases we maximize w.r.t. $\boldsymbol{\theta}$. Eqs. 11–13: Recursive DP equations for summing over $\boldsymbol{t}$ and $\boldsymbol{a}$.

alignments are treated as a hidden variable to be marginalized out.[10] Optimization problems of this form are by now widely known in NLP (Koo and Collins, 2005), and have recently been used for machine translation as well (Blunsom et al., 2008). Such problems are typically solved using variations of gradient ascent; in our experiments, we will use an online method called stochastic gradient ascent (SGA). This requires us to calculate the function's gradient (vector of first derivatives) with respect to $\boldsymbol{\theta}$.[11]

Computing the numerator in Eq. 9 involves summing over all possible alignments; with QDG and a hard bound of 1 on $|\boldsymbol{a}(j)|$ for all j, a fast "inside" DP solution is known (Smith and Eisner, 2006; Wang et al., 2007). It runs in $O(mn^2)$ time and $O(mn)$ space.

Computing the denominator in Eq. 9 requires summing over all word sequences and dependency trees for the target language sentence and all word alignments between the sentences. With a maximum length imposed, this is tractable using the "inside" version of the maximizing DP algorithm of Sec. 4, but it is prohibitively expensive. We therefore optimize *pseudo-likelihood* instead, making the following approximation (Besag, 1975):

$$p(\boldsymbol{t}, \tau_{\boldsymbol{t}} \mid \boldsymbol{s}, \tau_{\boldsymbol{s}}) \approx p(\boldsymbol{t} \mid \tau_{\boldsymbol{t}}, \boldsymbol{s}, \tau_{\boldsymbol{s}}) \times p(\tau_{\boldsymbol{t}} \mid \boldsymbol{t}, \boldsymbol{s}, \tau_{\boldsymbol{s}})$$

Plugging this into Eq. 9, we arrive at Eq. 10 (Tab. 3). The two parenthesized terms in Eq. 10 each have their own numerators and denominators (not shown). The numerators are identical to each other and to that in Eq. 9. The denominators are much more manageable than in Eq. 9, never requiring summation over more than two structures at a time. We must sum over target word sequences and word alignments (with fixed $\tau_{\boldsymbol{t}}$), and separately over target trees and word alignments (with fixed $\boldsymbol{t}$).

5.1 Summing over $\boldsymbol{t}$ and $\boldsymbol{a}$

The summation over target word sequences and alignments given fixed $\tau_{\boldsymbol{t}}$ bears a resemblance to the inside algorithm, except that the tree structure is fixed (Pereira and Schabes, 1992). Let $S(j, i, t)$ denote the sum of all translations rooted at position j in $\tau_{\boldsymbol{t}}$ such that $\boldsymbol{a}(j) = i$ and $t_j = t$.

Tab. 3 gives the equations for this DP: Eq. 11 is the quantity of interest, Eq. 12 is the recursion, and Eq. 13 shows the base cases for leaves of $\tau_{\boldsymbol{t}}$.

Letting $q = \max_{0 \le i \le n} |\mathrm{Trans}(s_i)|$, this algorithm runs in $O(mn^2q^2)$ time and $O(mnq)$ space. For efficiency we place a hard upper bound on q during training (details in §6).

5.2 Summing over $\tau_{\boldsymbol{t}}$ and $\boldsymbol{a}$

For the summation over dependency trees and alignments given fixed $\boldsymbol{t}$, required for $p(\tau_{\boldsymbol{t}} \mid \boldsymbol{t}, \boldsymbol{s}, \tau_{\boldsymbol{s}})$, we perform "inside" lattice parsing with $G_{\boldsymbol{s}, \tau_{\boldsymbol{s}}}$. The technique is the summing variant of the decoding method in §4, except for each state j,

[10] Alignments could be supplied by automatic word alignment algorithms. We chose to leave them hidden so that we could make the best use of our parsed training data when configuration constraints are imposed, since it is not always possible to reconcile automatic word alignments with automatic parses.

[11] When the function's value is computed by "inside" DP, the corresponding "outside" algorithm can be used to obtain the gradient. Because outside algorithms can be automatically derived from inside ones, we discuss only inside algorithms in this paper; see Eisner et al. (2005).

the sausage lattice only includes arcs from $j-1$ to j that are labeled with the known target word t_j. If a is the number of arcs in the lattice, which is $O(mn)$, this algorithm runs in $O(a^3)$ time and requires $O(a^2)$ space. Because we use a hard upper bound on $|\mathrm{Trans}(s)|$ for all $s \in \Sigma$, this summation is much faster in practice than the one over words and alignments.

5.3 Handling Non-Local Features

So far, all of our algorithms have exploited DP, disallowing any *non-local* features (e.g., $\boldsymbol{f}_{phr}$, $\boldsymbol{f}_N$ for $N > 1$, $\boldsymbol{f}_{\mathrm{zth}}$, $\boldsymbol{f}_{\boldsymbol{s}unc}$). We recently proposed "cube summing," an approximate technique that permits the use of non-local features for inside DP algorithms (Gimpel and Smith, 2009). Cube summing is based on a slightly less greedy variation of cube *pruning* (Chiang, 2007) that maintains k-best lists of derivations for each DP chart item. Cube summing augments the k-best list with a residual term that sums over remaining structures not in the k-best list, albeit without their non-local features. Using the machinery of cube summing, it is straightforward to include the desired non-local features in the summations required for pseudo-likelihood, as well as to compute their approximate gradients.

Our approach permits an alternative to minimum error-rate training (MERT; Och, 2003); it is discriminative but handles latent structure and regularization in more principled ways. The pseudo-likelihood calculations for a sentence pair, taken together, are faster than (k-best) decoding, making SGA's inner loop faster than MERT's inner loop.

6 Experiments

Our decoding framework allows us to perform many experiments with the same feature representation and inference algorithms, including combining and comparing phrase-based and syntax-based features and examining how isomorphism constraints of synchronous formalisms affect translation output.

6.1 Data and Evaluation

We use the German-English portion of the Basic Travel Expression Corpus (BTEC). The corpus has approximately 100K sentence pairs. We filter sentences of length more than 15 words, which only removes 6% of the data. We end up with a training set of 82,299 sentences, a development set of 934 sentences, and a test set of 500 sentences. We evaluate translation output using case-insensitive BLEU (Papineni et al., 2001), as provided by NIST, and METEOR (Banerjee and Lavie, 2005), version 0.6, with Porter stemming and WordNet synonym matching.

6.2 Features

Our base system uses features as discussed in §2. To obtain lexical translation features $\boldsymbol{g}_{trans}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{t})$, we use the Moses pipeline (Koehn et al., 2007). We perform word alignment using GIZA++ (Och and Ney, 2003), symmetrize the alignments using the "grow-diag-final-and" heuristic, and extract phrases up to length 3. We define $\boldsymbol{f}_{lex}$ by the lexical probabilities $p(t \mid s)$ and $p(s \mid t)$ estimated from the symmetrized alignments. After discarding phrase pairs with only one target-side word (since we only allow a target word to align to at most one source word), we define $\boldsymbol{f}_{phr}$ by 8 features: $\{2, 3\}$ target words $\times$ phrase conditional and "lexical smoothing" probabilities $\times$ two conditional directions.

Bigram and trigam language model features, $\boldsymbol{f}_2$ and $\boldsymbol{f}_3$, are estimated using the SRI toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing (Chen and Goodman, 1998).

For our target-language syntactic features $\boldsymbol{g}_{syn}$, we use features similar to lexicalized CFG events (Collins, 1999), specifically following the dependency model of Klein and Manning (2004). These include probabilities associated with individual attachments ($\boldsymbol{f}_{att}$) and child-generation valence probabilities ($\boldsymbol{f}_{val}$). These probabilities are estimated on the training corpus parsed using the Stanford factored parser (Klein and Manning, 2003). The same probabilities are also included using 50 hard word classes derived from the parallel corpus using the GIZA++ `mkcls` utility (Och and Ney, 2003). In total, there are 7 lexical and 7 word-class syntax features.

For reordering, we use a single absolute distortion feature $\boldsymbol{f}_{dist}(i, j)$ that returns $|i-j|$ whenever $\boldsymbol{a}(j) = i$ and $i, j > 0$. (Unlike the other feature functions, which returned probabilities, this feature function returns a nonnegative integer.)

The tree-to-tree syntactic features $\boldsymbol{g}_{tree^2}$ in our model are binary features $\boldsymbol{f}_{qg}$ that fire for particular QG configurations. We use one feature for each of the configurations in (Smith and Eisner, 2006), adding 7 additional features that score configura-

Phrase features:	Syntactic Features:		
	(base)	$+\boldsymbol{f}_{att} \cup \boldsymbol{f}_{val}$ (target)	$+\boldsymbol{f}_{qg}$ (tree-to-tree)
(base)	0.3727	0.4458	0.4424
$+\boldsymbol{f}_{phr}$	0.4682	0.4971	0.5142

Table 4: Feature set comparison (BLEU).

tions involving root words and NULL-alignments more finely. There are 14 features in this category.

Coverage features $\boldsymbol{g}_{cov}$ are as described in §4.2.

In all, 46 feature weights are learned.

6.3 Experimental Procedure

Our model permits training the system on the full set of parallel data, but we instead use the parallel data to estimate feature functions and learn $\boldsymbol{\theta}$ on the development set.[12] We trained using three iterations of SGA over the development data with a batch size of 1 and a fixed step size of 0.01. We used ℓ_2 regularization with a fixed, untuned coefficient of 0.1. Cube summing used a 10-best list for training and a 7-best list for decoding unless otherwise specified.

To obtain the translation lexicon (Trans) we first included the top three target words t for each s using $p(s \mid t) \times p(t \mid s)$ to score target words. For any training sentence $\langle \boldsymbol{s}, \boldsymbol{t} \rangle$ and t_j for which $t_j \notin \bigcup_{i=1}^{n} \text{Trans}(s_i)$, we added t_j to $\text{Trans}(s_i)$ for $i = \text{argmax}_{i' \in I}\, p(s_{i'}|t_j) \times p(t_j|s_{i'})$, where $I = \{i : 0 \leq i \leq n \wedge |\text{Trans}(s_i)| < q_i\}$. We used $q_0 = 10$ and $q_{>0} = 5$, restricting $|\text{Trans}(\text{NULL})| \leq 10$ and $|\text{Trans}(s)| \leq 5$ for any $s \in \Sigma$. This made 191 of the development sentences unreachable by the model, leaving 743 sentences for learning $\boldsymbol{\theta}$.

During decoding, we generated lattices with all $t \in \text{Trans}(s_i)$ for $0 \leq i \leq n$, for every position. We used $\rho = 0.9$, causing states within 90% of the source sentence length to be final states. Between each pair of consecutive states, we pruned edges that fell outside a beam of 70% of the sum of edge weights (see §4.1; edge weights use $\boldsymbol{f}_{lex}$, $\boldsymbol{f}_{dist}$, and $\boldsymbol{f}_{\boldsymbol{s}cov}$) of all edges between those two states.

6.4 Feature Set Comparison

Our first set of experiments compares feature sets commonly used in phrase- and syntax-based translation. In particular, we compare the effects of combining phrase features and syntactic features. The base model contains $\boldsymbol{f}_{lex}$, $\boldsymbol{g}_{lm}$, $\boldsymbol{g}_{reor}$, and $\boldsymbol{g}_{cov}$. The results are shown in Table 4. The second row contains scores when adding in the eight $\boldsymbol{f}_{phr}$ features. The second column shows scores when adding the 14 target syntax features ($\boldsymbol{f}_{att}$ and $\boldsymbol{f}_{val}$), and the third column adds to them the 14 additional tree-to-tree features ($\boldsymbol{f}_{qg}$). We find large gains in BLEU by adding more features, and find that gains obtained through phrase features and syntactic features are partially additive, suggesting that these feature sets are making complementary contributions to translation quality.

6.5 Varying k During Decoding

For models without syntactic features, we constrained the decoder to produce dependency trees in which every word's parent is immediately to its right and ignored syntactic features while scoring structures. This causes decoding to proceed left-to-right in the lattice, the way phrase-based decoders operate. Since these models do not search over trees, they are substantially faster during decoding than those that use syntactic features and do not require any pruning of the lattice. Therefore, we explored varying the value of k used during k-best cube decoding; results are shown in Fig. 2. Scores improve when we increase k up to 10, but not much beyond, and there is still a substantial gap (2.5 BLEU) between using phrase features with $k = 20$ and using all features with $k = 5$. Models without syntax perform poorly when using a very small k, due to their reliance on non-local language model and phrase features. By contrast, models with syntactic features, which are local in our decoder, perform relatively well even with $k = 1$.

6.6 QG Configuration Comparison

We next compare different constraints on isomorphism between the source and target dependency

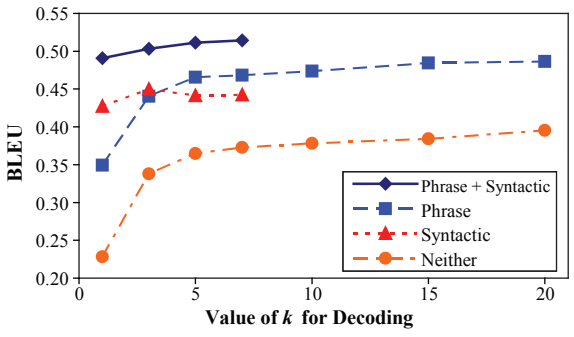


Figure 2: Comparison of size of k-best list for cube decoding with various feature sets.

[12] We made this choice both for similarity to standard MT systems and a more rapid experiment cycle.

QDG Configurations	BLEU	METEOR
synchronous	0.4008	0.6949
+ nulls, root-any	0.4108	0.6931
+ child-parent, same node	0.4337	0.6815
+ sibling	0.4881	0.7216
+ grandparent/child	0.5015	0.7365
+ c-command	0.5156	0.7441
+ other	0.5142	0.7472

Table 5: QG configuration comparison. The name of each configuration, following Smith and Eisner (2006), refers to the relationship between $\boldsymbol{a}(\tau_{\boldsymbol{t}}(j))$ and $\boldsymbol{a}(j)$ in $\tau_{\boldsymbol{s}}$.

trees. To do this, we impose harsh penalties on some QDG configurations (§3) by fixing their feature weights to -1000. Hence they are permitted only when absolutely necessary in training and rarely in decoding.[13] Each model uses all phrase and syntactic features; they differ only in the sets of configurations which have fixed negative weights.

Tab. 5 shows experimental results. The base "synchronous" model permits parent-child ($\boldsymbol{a}(\tau_{\boldsymbol{t}}(j)) = \tau_{\boldsymbol{s}}(\boldsymbol{a}(j))$), any configuration where $\boldsymbol{a}(j) = 0$, including both words being linked to NULL, and requires the root word in $\tau_{\boldsymbol{t}}$ to be linked to the root word in $\tau_{\boldsymbol{s}}$ or to NULL(5 of our 14 configurations). The second row allows any configuration involving NULL, including those where t_j aligns to a non-NULL word in $\boldsymbol{s}$ and its parent aligns to NULL, and allows the root in $\tau_{\boldsymbol{t}}$ to be linked to any word in $\tau_{\boldsymbol{s}}$. Each subsequent row adds additional configurations (i.e., trains its θ rather than fixing it to -1000). In general, we see large improvements as we permit more configurations, and the largest jump occurs when we add the "sibling" configuration ($\tau_{\boldsymbol{s}}(\boldsymbol{a}(\tau_{\boldsymbol{t}}(j))) = \tau_{\boldsymbol{s}}(\boldsymbol{a}(j))$). The BLEU score does not increase, however, when we permit all configurations in the final row of the table, and the METEOR score increases only slightly. While allowing certain categories of non-isomorphism clearly seems helpful, permitting arbitrary violations does not appear to be necessary for this dataset.

6.7 Discussion

We note that these results are not state-of-the-art on this dataset (on this task, Moses/MERT achieves 0.6838 BLEU and 0.8523 METEOR with maximum phrase length 3).[14] Our aim has been to illustrate how a single model can provide a controlled experimental framework for comparisons of features, of inference methods, and of constraints. Our findings show that phrase features and dependency syntax produce complementary improvements to translation quality, that tree-to-tree configurations (a new feature in MT) are helpful for translation, and that substantial gains can be obtained by permitting certain types of non-isomorphism. We have validated cube summing and decoding as practical methods for approximate inference.

Our framework permits exploration of alternative objectives, alternative approximate inference techniques, additional hidden variables (e.g., Moses' phrase segmentation variable), and, of course, additional feature representations. The system is publicly available at `www.ark.cs.cmu.edu/Quipu`.

7 Conclusion

We presented feature-rich MT using a principled probabilistic framework that separates features from inference. Our novel decoder is based on efficient DP-based QG lattice parsing extended to handle "non-local" features using generic techniques that also support efficient parameter estimation. Controlled experiments permitted with this system show interesting trends in the use of syntactic features and constraints.

Acknowledgments

We thank three anonymous EMNLP reviewers, David Smith, and Stephan Vogel for helpful comments and feedback that improved this paper. This research was supported by NSF IIS-0836431 and IIS-0844507, a grant from Google, and computational resources provided by Yahoo.

References

H. Alshawi, S. Bangalore, and S. Douglas. 2000. Learning dependency translation modles as colections of finite-state head transducers. *Computational Linguistics*, 26(1):45–60.

S. Banerjee and A. Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*.

J. E. Besag. 1975. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195.

[13] In fact, the strictest "synchronous" model used the almost-forbidden configurations in 2% of test sentences; this behavior disappears as configurations are legalized.

[14] We believe one cause for this performance gap is the generation of the lattice and plan to address this in future work by allowing the phrase table to inform lattice generation.

P. Blunsom and M. Osborne. 2008. Probabilistic inference for machine translation. In *Proc. of EMNLP*.

P. Blunsom, T. Cohn, and M. Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proc. of ACL*.

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

S. Chen and J. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report 10-98, Harvard University.

D. Chiang, Y. Marton, and P. Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proc. of EMNLP*.

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

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, U. Penn.

D. Das and N. A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proc. of ACL-IJCNLP*.

Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammar. In *Proc. of ACL*.

J. Eisner, E. Goldlust, and N. A. Smith. 2005. Compiling Comp Ling: Practical weighted dynamic programming and the Dyna language. In *Proc. of HLT-EMNLP*.

J. Eisner. 1997. Bilexical grammars and a cubic-time probabilistic parser. In *Proc. of IWPT*.

M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, and I. Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of COLING-ACL*.

K. Gimpel and N. A. Smith. 2008. Rich source-side context for statistical machine translation. In *Proc. of ACL-2008 Workshop on Statistical Machine Translation*.

K. Gimpel and N. A. Smith. 2009. Cube summing, approximate inference with non-local features, and dynamic programming without semirings. In *Proc. of EACL*.

R. Haque, S. K. Naskar, Y. Ma, and A. Way. 2009. Using supertags as source language context in SMT. In *Proc. of EAMT*.

L. Huang and D. Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proc. of ACL*.

A. Ittycheriah and S. Roukos. 2007. Direct translation model 2. In *Proc. of HLT-NAACL*.

D. Klein and C. D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in NIPS 15*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of ACL*.

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

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL (demo session)*.

T. Koo and M. Collins. 2005. Hidden-variable models for discriminative reranking. In *Proc. of EMNLP*.

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of COLING-ACL*.

A. Lopez. 2009. Translation as weighted deduction. In *Proc. of EACL*.

D. Marcu, W. Wang, A. Echihabi, and K. Knight. 2006. Statistical machine translation with syntactified target language phrases. In *Proc. of EMNLP*.

H. Mi, L. Huang, and Q. Liu. 2008. Forest-based translation. In *Proc. of ACL*.

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

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

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

K. Papineni, S. Roukos, and T. Ward. 1997. Feature-based language understanding. In *EUROSPEECH*.

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

F. C. N. Pereira and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proc. of ACL*.

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

L. Shen, J. Xu, and R. Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL*.

D. A. Smith and J. Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proc. of HLT-NAACL Workshop on Statistical Machine Translation*.

D. A. Smith and J. Eisner. 2009. Parser adaptation and projection with quasi-synchronous features. In *Proc. of EMNLP*.

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

X. Sun and J. Tsujii. 2009. Sequential labeling with latent variables: An exact inference algorithm and its efficient approximation. In *Proc. of EACL*.

M. Wang, N. A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proc. of EMNLP-CoNLL*.

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

Improved Word Alignment with Statistics and Linguistic Heuristics

Ulf Hermjakob
University of Southern California
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292, USA
ulf@isi.edu

Abstract

We present a method to align words in a bitext that combines elements of a traditional statistical approach with linguistic knowledge. We demonstrate this approach for Arabic-English, using an alignment lexicon produced by a statistical word aligner, as well as linguistic resources ranging from an English parser to heuristic alignment rules for function words. These linguistic heuristics have been generalized from a development corpus of 100 parallel sentences. Our aligner, UALIGN, outperforms both the commonly used GIZA++ aligner and the state-of-the-art LEAF aligner on F-measure and produces superior scores in end-to-end statistical machine translation, +1.3 BLEU points over GIZA++, and +0.7 over LEAF.

1 Introduction

Word alignment is a critical component in training statistical machine translation systems and has received a significant amount of research, for example, (Brown et al., 1993; Ittycheriah and Roukos, 2005; Fraser and Marcu, 2007), including work leveraging syntactic parse trees, e.g., (Cherry and Lin, 2006; DeNero and Klein, 2007; Fossum et al., 2008). Word alignment is also a required first step in other algorithms such as for learning sub-sentential phrase pairs (Lavie et al., 2008) or the generation of parallel treebanks (Zhechev and Way, 2002).

Yet word alignment precision remains surprisingly low, under 80% for state-of-the-art aligners on not closely related language pairs.

Consider the following Arabic/English sentence pair with alignments built by the statistical word aligner LEAF:

Bitext Arabic: وفاز التايلاندى بارادورن سريشافان على الاسترالى جايسون ستولتنبرغ 4 - 6 و 4 - 6 ,والتشيكى ييرى فانيك على الالمانى لارش بورغسمولر 4 - 6 و 4 - 6

Gloss: Won(1) Thai Paradorn Srichaphan(1) on/to(2) Australian Jason(2) Stoltenberg(3) 6(4) - 4(5) and(3) 6(4) - 4(5), and Czech Jiří(7) Vaněk(7) on/to German(6) Lars Burgsmüller 6(4) - 4 and(3) 6(4) - 4

Bitext English: Thailand 's(1) *Baradorn Srichfan*(1) beat(2) Australian *Gayson*(1) *Stultenberg*(3) 6(4) - *6*(4) 6(4) - 4(5) , Czech player(1) *Pierre*(1) *Vanic*(7) beat(6) Germany(6) 's Lars Burgsmuller 6 - 4 6 - 4

In the example above, words with the same index in the gloss for Arabic and the English are aligned to each other, alignment errors are underlined, translation errors are in *italics*. For example, the Arabic words for *won* and *Srichaphan* are aligned with the English words *'s*, *Srichfan*, *Gayson*, *player* and *Pierre*.

As reflected in the example above, typical alignment problems include

- words that change sentence position between languages, such as verbs, which in Arabic are often sentence-initial (e.g. *won/beat* in the example above)
- function words without a clear and explicit equivalent in the other language (e.g. the Arabic و/*and* in the example above)
- lack of robustness with respect to poor translations (e.g. *Gayson Stultenberg* instead of *Jason Stoltenberg*) or bad sentence alignment.

We believe we can overcome such problems with the increased use of linguistically based

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 229–237,
Singapore, 6-7 August 2009.

heuristics. We can model typical word order differences between English and Arabic using English parse trees and a few Arabic-specific phrase reordering heuristics. We can narrow the space of possible alignment candidates for function words using English parse trees and a few heuristics for each type of function word.

These heuristics have been developed using a development corpus of 100 parallel sentences. The heuristics are generalizations based on patterns of misaligned words, misaligned with respect to a Gold Standard alignment for that development corpus.

The following sections describe how our word aligner works, first how relatively reliable content words are aligned, and then how function words and any remaining content words are aligned, with a brief discussion of an interesting issue relating to the Gold Standard we used. Finally we present evaluations on word alignment accuracy as well as the impact on end-to-end machine translation quality.

2 Phase I: Content Words

We divide the alignment process into two phases: first, we align relatively reliable content words, which in phase II we then use as a skeleton to align function words and remaining content words.

Function words such as English *a, ah, all, am, an, and, any, are, as, at, ...* are common words that often do not have an explicit equivalent word or words in the other side of the bitext. In our system, we use a list of 96 English and 110 Arabic function words with those characteristics. For the purposes of our algorithm, a word is a function word if and only if it is on the function word list for its language. A content word then is defined as a word that is neither a function word nor punctuation.

The approach for aligning content words in phase I is as follows: First, we score each combination of an Arabic content word and English content word in an aligned sentence and align those pairs that pass a threshold, typically generating too many alignments. Second, we compute a more comprehensive score that also takes into consideration matching alignments in the context around each alignment. Third, we eliminate inferior alignments that are incompatible with higher-scoring alignments.

The score in the first step is *pointwise mutual information* (PMI). The key resource to compute this PMI is an alignment lexicon generated beforehand by a statistical word alignment system from a large bitext. An alignment lexicon is a list of triples, each consisting of an English word, an Arabic word, and how often they have been aligned for a given bitext. Additional counts on how often each English and Arabic word occurs allow us use this alignment lexicon to compute PMI(e,f) = $\log \frac{p(e,f)}{p(e) \cdot p(f)}$. We align those Arabic and English content words that have a PMI > 0 and a minimum alignment lexicon count (≥ 10 initially). Using the alignment lexicon generated by a statistical word aligner to compute PMIs is the principal statistical component in our system. We explored alternative metrics such as the dice-coefficient that was used by other researchers in earlier alignment work, but found PMI to work better for our system.

In a second step, we lay a window of size 5 around each aligned pair of Arabic and English words (counting only content words) and then add to the PMI score of the link itself the PMI scores of other links within that window, with a distance weight of $\frac{1}{distance+1}$. This yields a new score that takes into account whether a link is supported by context.

In the third step, we check for overgenerated links, comparing links that share an Arabic or an English word. If a word on one side of the bitext is linked to multiple *adjacent* words on the other, we leave them alone, as one word in one language often corresponds to multiple words in the other. However, if a word on one side is linked to non-adjacent words on the other side, this flags an incompatibility, and we remove those links that have inferior context-sensitive scores. This removal is done one link at a time, with the lowest relative scores first.

We boost the process we just described in a few ways. In the first alignment step, we also include as alignment candidates any content words that are string-identical on each side, such as ASCII numbers and ASCII words. We finally also include as alignment candidates those word pairs that are transliterations of each other to cover rare proper names (Hermjakob et al., 2008), which is important for language pairs that don't share the same alphabet such as Arabic and English.

2.1 Reordering Using an English Parser

We use a refined notion of context window that models word order differences between Arabic and English. Traversing a parse tree for English, we identify sub-trees for which the order in Arabic can be substantially different. In Arabic, for example, the verb is often sentence-initial. So for trees or subtrees identified by the parser as sentences, we generate an alternative reordering of its subtrees where the verb has been moved to the front. Similarly, in a noun phrase, we generate an alternative order where adjectives are moved to the right of the noun they modify.

For example, consider the sentence *John bought a new car* . We can reorder its parse tree both at the sentence level: *(bought) (John) (a new car) (.)* as well as at its object NP level: *(a) (car) (new)*. If fully enumerated, this would yield these four reordering alternatives:

1. John bought a new car .
2. John bought a car new .
3. bought John a new car .
4. bought John a car new .

We don't actually explicitly enumerate all variants but keep all reordering alternatives in a reordering forest, since the number of fully expanded reorderings grows exponentially with the number of phrases with reordering(s). At the beginning of Phase I, we compute from this reordering forest a minimum distance matrix, which, for specific instances of the words *John* and *car* would record a minimum distance of 1 (based on reordering 4, skipping the function word *a*).

For the example sentence at the beginning of the paper we would get reorderings including the following:

Engl. orig.: thailand 's baradorn srichfan beat ...
A reordering: beat thailand 's baradorn srichfan ...
Arabic (gloss): won thai paradorn srichaphan ...

In the above reordered English alternative, *beat* and *thailand* are next to each other, so their minimum distance is 1, which means that a link between English *thailand* and Arabic *thai* now strongly boosts the context-sensitive score between English *beat* and Arabic *won*.

2.2 Morphological Variation

Another challenge to content word alignment is morphological variation which can create data sparsity in the alignment lexicon. For example, in a given bitext sentence, the Arabic word *AlAwDAE* might be translated as *situational*, for which there might be no support in the alignment lexicon. However the PMI between *AlAwDAE* and *situation* might be sufficiently high. Additionally, there is another Arabic word, *AlHAlAt*, which often translates as both *situation* and *situational*.

To take advantage of such constellations, we built morphological variation lists for both Arabic and English, lists that for a given head word such as *situational* lists variants such as *situation*, and *situations*.

We built these lists in a one-time process by identifying superficially similar words, i.e. those that vary only with respect to an ending or a prefix, and then semantically validating such candidates using a pivot word in the other language such as *AlHAlAt* that has sufficiently strong alignment lexicon co-alignment counts with both *situation* and *situational*. The alignment lexicon co-alignment count of an Arabic word w_{ar} and an English word w_{en} is considered strong enough, if it is at least 2.0 and at least 0.001 times as high as the highest co-alignment count of w_{ar} with any English word; words shorter than four letters are excluded from consideration. So because *situation* and *situational* are superficially similar **and** they are both have a strong alignment count with *AlHAlAt* in the alignment lexicon, *situation* is added to the English morphological variation list as a variant of *situational* and vice versa.

Exploring whether we can align *situational* and *AlAwDAE* in the bitext, we find that *situational* is a morphological variant of *situation* (based on our morphological variation list for English); next we find that based on the alignment lexicon, there is a positive PMI between *situation* and *AlAwDAE*, which completes the chain between *situational* and *AlAwDAE*, so we include them as an alignment candidate after all. The PMI of such a morphological-variation-based candidate is weighted by a 'penalty' factor of 0.5 when compared with the PMI of any competing alignment candidate without such morphological-variation step.

Similarly, the English pivot word *situations* can be used to semantically validate the similarity between Arabic *AlAwDAE* and *AwDAE* for our Arabic morphological variation list. The resulting Arabic morphological variation list has entries for 193,263 Arabic words with an average of 4.2 variants each; our English morphological variation list has 57,846 entries with 2.8 variants

each.

At the end of phase I, most content words will be aligned with relatively high precision. Since function words often do not have an explicit equivalent word or words in the other side of a bitext, they can not be aligned as reliably as content words based on bilingual PMI.[1] Note that due to data sparsity, some content words will remained unaligned in phase I and will subsequently be aligned in phase II as explained in section 3.3.

3 Phase II: Function Words

In Phase II, we align function words, punctuation, and some remaining content words. Function words can be classified into three categories: monovalent, divalent and independent. Monovalent function words modify one head; they include articles (which modify nouns), possessive pronouns, demonstrative adjectives and auxiliary verbs. Divalent function words connect two words or phrases; they include conjunctions and prepositions. Independent function words include non-possessive pronouns and copula (e.g. *is* as a main verb). Each of these types of function words is aligned according to its own heuristics.

In this section we present three representative examples, one for articles (monovalent), one for prepositions (divalent), as well as a structural heuristic.

3.1 Example: Articles

Monovalent function words have the simplest heuristics. Recall that Arabic does not have articles (only a definite prefix *Al-* added to one or more words in a definite noun phrase), so there is usually no explicit equivalent of the English article on the Arabic side.

For an English article, our system identifies the English head word that it modifies based on the English parse tree, and then aligns it with the same Arabic word(s) which that head word is aligned with.

3.2 Example: Prepositions

Divalent function words are much more interesting. In many cases, an English preposition corresponds to an explicit Arabic preposition in basically the same position. Alignment in that case is straightforward. However, some Arabic prepositions and even more English prepositions do not have an explicit counterpart on the other side. We call such prepositions *orphan prepositions*. The English preposition *of* is almost always orphaned in this way.

The decision how to align such an orphan preposition is not trivial. Consider the bitext *island of Basilan/jzyrp bAsylAn*, a typical (NP1 (P NP2)) construction on the English side. Should we co-align the preposition *of* with the head of NP1 or the head of NP2? In English syntax, the preposition is grouped with NP2, but a preposition is often better "motivated" by NP1. We therefore decided to use the English parse tree to identify the heads of both NP1 and NP2, identify the Arabic words aligned to these heads as candidates, and then align the preposition to the Arabic candidate word with which it has the highest bilingual PMI. It turns out that in most cases this will be the candidate on the "left". For the example at the top of this paragraph, *of* will be aligned with *jzyrp* ("island"), which is actually desirable for MT, as it facilitates subsequent rule extraction of type "island of X/jzyrp X". We refer to this orphan preposition alignment style as *MT-style*.

According to the gold standard alignment guidelines used for the LDC Gold Standard however, an orphan preposition should always be aligned to the "right", to *bAsylAn* in the example above. We therefore implemented an alternative *GS-style* (for "Gold Standard") to be able to later evaluate the impact of these alternatives alignment styles.

The question whether GIZA or LEAF alignments will indeed give meaningful scores to support the *MT-style* attachments will be answered by the MT experiments described in section 4.3.

Here is a more complex example with Arabic (A), its gloss (G) and English (E):

Arabic: الاحد اغارت الطائرات الاميركية على منطقة جوار
Gloss: sunday attacked aircraft american on/to area jiwar
Engl.: on sunday american aircraft attacked the area of jiwar

For the Arabic orphan preposition على/*ElY* ("on/to"), our system identifies two candidates based on the English parse tree: *attacked* and *area*. Based on a higher mutual information, our system then aligns Arabic *ElY* ("on/to") with English *attacked*, which results in the English word *attacked* now being aligned to both Arabic *attacked* and the

[1] It is this lack of reliability that is the defining characteristic of our function words, differentiating them from the concept of marker words used in EBMT chunking (Way and Gough, 2002).

Arabic *on/to*, even though they are not adjacent. In the Gold Standard, Arabic *on/to* is aligned with English *area*, and LEAF aligns it with English *on* (yes, the one preceding Sunday). This is apparently very tempting as Arabic *on/to* is often translated as English *on*, but here it is incorrect, and our system avoids this tempting alignment because it is ruled out linguistically.

Note that in some cases, such as sentence-initial prepositional phrases, there is only one candidate; occasionally, when relevant content words remain unaligned, no candidate can be identified, in which case the orphan preposition remains unaligned as well.

3.3 Example: Adjectives

It is not uncommon that content words that we would like to be aligned are not supported by the alignment lexicon, due to general data sparsity or maybe a somewhat unorthodox translation. In those cases we can use structure and word order knowledge to make reasonable alignments anyway.

Consider an English noun phase ADJ-E NOUN-E and the corresponding Arabic NOUN-A ADJ-A. If the nouns are already aligned, but the adjectives are not yet aligned, we can use the English parse tree to identify ADJ-E as a modifier to NOUN-E, and, aware that adjectives in Arabic post-modify their nouns, identify the corresponding Arabic word based on structure and word order alone. This can be done the other way around as well (link nouns based on already aligned adjectives) and other elements of other phrases as well.

As more and more function words and remaining content words get aligned, heuristics that weren't applicable before may now apply to the remaining unaligned words, so we perform four passes through a sentence pair to align unaligned words using heuristics. We found that an additional fifth pass did not yield any further improvements.

4 Experiments

We evaluated our word aligner in terms of both alignment accuracy and its impact on an end-to-end machine translation system.

4.1 Alignment Experiments

We evaluated our word aligner against a Gold Standard distributed by LDC. The human alignments of the sentences in this Gold Standard are based on the 2006 GALE Guidelines for Arabic Word Alignment Annotation.

Both the 100-sentence development set and the separate 837-sentence test set are Arabic newswire sentences from LDC2006E86. The test set includes only sentences for which our English parser (Soricut and Marcu, 2003) could produce a parse tree, which effectively excluded a few very long sentences.

In the first set of experiments, we compare two settings of our UALIGN system with other aligners, GIZA++ (Union) (Och and Ney, 2003) and LEAF (with 2 iterations) (Fraser and Marcu, 2007). The GIZA++ aligner is based on IBM Model 4 (Brown et al., 1993). We chose GIZA Union for our comparison, because it led to a higher BLEU score for our overall MT system than other GIZA variants such as GIZA Intersect and Grow-Diag. The two settings of our system vary in the style on how to align orphan prepositions. Besides precision, recall and (balanced) F-measure, we also include an F-measure variant strongly biased towards recall (α=0.1), which (Fraser and Marcu, 2007) found to be best to tune their LEAF aligner for maximum MT accuracy. GIZA++ and LEAF alignments are based on a parallel training corpus of 6.6 million sentence pairs, incl. the LDC2006E86 set mentioned above.

Aligner	Prec.	Recall	F-0.5	F-0.1
GIZA	26.9	84.3	40.8	69.5
LEAF	73.3	79.7	76.4	79.0
UALIGN MT-style	82.5	80.0	81.2	80.2
UALIGN GS-style	84.0	82.9	83.5	83.0

Table 1: Alignment precision, recall, F-measure (α=0.5), F-measure(α=0.1) for different aligners; with UALIGN using LEAF alignment lexicon.

Our aligner outperforms both GIZA and LEAF on all metrics. Not surprisingly, the GS-style alignments, which align "orphan" prepositions according to Gold Standard guidelines, yield higher scores than MT-style alignments. And interestingly by a remarkably high margin.

In a second set of experiments, we measure the impact of using different input alignment lexicon used by our aligner on alignment accuracy. In one case UALIGN uses as input the alignment lexicon produced by LEAF, in the other the alignment lexicon produced by GIZA. All experiments in table 2

are for UALIGN.

Style	A-Lexicon	Prec.	Recall	F-0.5	F-0.1
MT	from LEAF	82.5	80.0	81.2	80.2
MT	from GIZA	80.8	79.2	80.0	79.4
GS	from LEAF	84.0	82.9	83.5	83.0
GS	from GIZA	82.1	81.8	82.0	81.9

Table 2: Alignment precision, recall, F-measure (α=0.5), F-measure(α=0.1), all of UALIGN, for different alignment styles, different input alignment lexicons.

As LEAF clearly outperforms GIZA on F-0.1 (79.0 vs. 69.5, see table 1), the alignment lexicon based on LEAF is better, so it is not surprising that when we use an alignment lexicon based on GIZA, all metrics degrade, and consistently so for both alignment styles. However the drop in F-0.1 of about 1 point (80.2 $\rightarrow$ 79.4 and 83.0 $\rightarrow$ 81.9) is much smaller than the differences between the underlying aligners themselves. Our aligner therefore degrades quite gracefully for a worse alignment lexicon.

Aligner	Arabic aligned	Engl. aligned
GIZA Union	100%	100%
LEAF	99.99%	97.25%
UALIGN	92.10%	91.55%
Gold Standard	95.37%	95.86%

Table 3: Percentages of Arabic and English words aligned

Table 3 shows how much LEAF and UALIGN differ in the percentage of Arabic and English words aligned (correctly or incorrectly). LEAF is much more aggressive in making alignments, aligning almost every Arabic word. Our aligner still leaves some 8% of all words in a sentence unaligned (an opportunity for further improvements). For comparison, in the Gold Standard, 4-5% of all words in our test corpus are left unaligned.

4.2 Impact of Sub-Components

To better understand the impact of several alignment system sub-components, we ran a number of experiments disabling individual sub-components and then comparing the resulting alignment scores with those of the full system. We also measured alignment scores running Phase II with 0 to 5 passes. The test set was the same as in section 4.1.

System	Prec.	Recall	F-0.1
Full system (FS)	84.0	82.9	83.0
FS w/o morph.variation	84.0	82.4	82.5
FS w/o Engl. tree reord.	83.8	82.7	82.8
FS w/o string identity	84.0	82.8	82.9
FS w/o name translit.	84.0	82.8	82.9
System after Phase I	90.6	44.5	46.8
+ Phase II w/ 1 pass	87.6	77.1	78.0
+ Phase II w/ 2 passes	85.8	80.3	80.8
+ Phase II w/ 3 passes	84.2	82.7	82.8
+ Phase II w/ 4 passes	84.0	82.9	83.0
+ Phase II w/ 5 passes	84.0	82.9	83.0

Table 4: Impact of sub-components on alignment precision, recall, F-measure, with GS-style attachments, based on the LEAF alignment lexicon.

Special sub-components of Phase I include adding link candidates for ASCII-string-identical words and transliterated names (see last paragraph before section 2.1), reordering using an English parser (section 2.1) and morphological variation (section 2.2). Each of these sub-components provides a small boost to F-0.1, ranging from +0.1 to +0.5. The second part of the table shows alignment scores before and after each pass of Phase II. Our full system includes 4 passes; an additional 5th pass did not yield any further improvements. Note that during Phase II, precision drops. This is a reflection of (1) our strategy to first align relatively reliable content words in Phase I, followed by less reliable function words and remaining content words, and (2) the challenges of building reliable Gold Standard alignments for function words and non-literal translations.

4.3 MT Experiments

The ultimate test for a word aligner is to measure its impact on an end-to-end machine translation system. For this we aligned 170,863 pairs of Arabic/English newswire sentences from LDC, trained a state-of-the-art syntax-based statistical machine translation system (Galley et al., 2006) on these sentences and alignments, and measured BLEU scores (Papineni et al., 2002) on a separate set of 1298 newswire test sentences. Besides swapping in a new set of alignments for the same set of training sentences, and automatically retuning the parameters of the translation system for each set of alignments, no other changes or adjustments were made to the existing MT system.

In the first set of experiments, we compare two settings of our UALIGN system with other aligners, again GIZA++ (Union) and LEAF (with 2 iterations). The two settings vary in the alignment lexicon that the UALIGN aligner uses as input.

Aligner	BLEU
GIZA	47.4
LEAF	48.0
UALIGN using GIZA alignment-lexicon	48.4
UALIGN using LEAF alignment-lexicon	48.7

Table 5: BLEU scores in end-to-end statistical MT system based on different aligners. Both UALIGN variants use MT-style alignments.

With a BLEU score of 48.7, UALIGN using a LEAF alignment-lexicon is significantly better than both GIZA (+1.3) and LEAF (+0.7). This and other significance assertions in this paper are based on paired bootstrap resampling tests with 95% confidence. UALIGN using a GIZA alignment-lexicon significantly outperforms GIZA itself (+1.0).

In a second experiment, we measured the impact of the two alignment styles on BLEU. Recall that for GS-style alignments, orphan prepositions are always co-aligned to the right, following Gold Standard annotation guidelines, whereas for MT-style alignments, mutual information is used to decide whether to align orphan prepositions to the left or to the right.

Aligner	BLEU
LEAF	48.0
UALIGN with GS-style alignments	48.0
UALIGN with MT-style alignments	48.7

Table 6: BLEU scores in end-to-end statistical MT system based on different alignment styles for orphan prepositions. Both UALIGN variants use a LEAF alignment lexicon.

While the GS-style alignments yielded a 2.8 point higher F-0.1 score (83.0 vs. 80.2), the MT-style alignments result in a significantly better BLEU score (48.7 vs. 48.0). This shows that (1) a seemingly small difference in alignment styles can have a remarkably high impact on both BLEU scores and alignment accuracy as measured against a Gold Standard, and that (2) optimizing alignment accuracy against an alignment Gold Standard does **not** necessarily optimize BLEU in end-to-end MT. The latter has been observed by other researchers before, but these results additionally suggest that the gold-standard annotation style might itself have to shoulder part of the blame.

4.4 Corpus Noise Robustness

In a small random "sanity check" sample from the 170,863 training sentences for the MT experiment, we found cases where the sentence in one language contained much more material than the sentence in the other language. Consider, for example the following sentence pair (with spurious material underlined):

Arabic:
لكن ايضا هناك بند اخر ينص على انه اذا لم ينشا الفندق ،

Gloss: but also there-is clause another stipulates on/to that if not established the-hotel ,

English: but , also there is another clause that stipulates that if the hotel is not established , then the government shall be compensated .

Both LEAF and UALIGN correctly align the English "*but , also ... not established ,*" with the Arabic side. LEAF further aligns all words in the spurious English "*then the government shall be compensated .*" with seemingly random material on the Arabic side, whereas UALIGN leaves these spurious words completely unaligned. It would be reasonable to speculate that this behavior, observed in several cases, may be contributing to the good BLEU scores.

5 Discussion

Building on existing statistical aligners, our new word aligner significantly outperforms the best word aligner to date in both alignment error rate and BLEU score.

We have developed an approach to word alignment that combines a statistical component with linguistic heuristics. It is novel in that it goes beyond generic resources such as parsers, adding heuristics to explicitly model word order differences and function word alignment.

The approach has numerous benefits. Our system produces superior results both on alignment accuracy and end-to-end machine translation quality. Alignments have a high precision. The system is fast (about 0.7 seconds per sentence), and sentences are aligned individually so that a large corpus can easily be aligned on several computers in

parallel. All alignment links are tagged with additional information, such as which phase and/or heuristic created them, yielding extensive explanatory power to the developer for easy understanding on how the system arrived at a given alignment. Our approach needs and uses a parser for only one side (English) and not for the other (Arabic).

On the other hand, some of the components of this aligner are language-specific, such as word order heuristics, the list of specific function words, and morphological variation lists. While these parts of the system need to be adapted for new languages, the overall architecture and types of heuristics and function words are language-independent. Chinese for example has different specific types of function words such as aspect markers and measure words. But these fall into the existing category of monovalent function words and will be treated according the same principles as other monovalent function words (section 3.1). Similarly, Japanese postpositions would be treated like other divalent function words (such as Arabic or English prepositions). The author and developer has a basic knowledge of Arabic in general, and an intermediate knowledge of Arabic grammar, which means that no intimate knowledge of Arabic was required to develop the language-specific components. This same author and developer recently started to adapt UALIGN to Chinese-English word alignment.

The alignment rate is still somewhat low. We plan to increase it by enlarging our development set beyond 100 sentences and adding further heuristics, as well as generalizing the output word alignment structure to allow alignments of words to larger constituents in a tree, and to explicitly assert that some words are not covered by the other side of a bitext to model poor translations and poor sentence alignments.

Acknowledgment

This research was supported under DARPA Contract No. HR0011-06-C-0022. The author would like to thank Kevin Knight and the anonymous reviewers for their helpful suggestions, and Steve DeNeefe for running the end-to-end MT evaluations.

References

Peter E. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. In *Computational Linguistics* Vol. 19(2), pages 263–311.

Colin Cherry and Dekang Lin. 2006. Soft Syntactic Constraints for Word Alignment Through Discriminative Training. In *Proceedings of the 44th Annual Meeting on Association for Computational Linguistics*, Sydney, Australia, pages 105–112.

John DeNero and Dan Klein. 2007. Tailoring Word Alignments to Syntactic Machine Translation. In *Proceedings of the 45th Annual Meeting on Association for Computational Linguistics*, Prague, Czech Republic, pages 17–24.

Victoria Fossum, Kevin Knight and Steven Abney. 2008. Using Syntax to Improve Word Alignment Precision for Syntax-Based Machine Translation. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, Columbus, Ohio, pages 44–52.

Alexander Fraser and Daniel Marcu. 2007. Getting the Structure Right for Word Alignment: LEAF. In *Proceedings of Conference for Empirical Methods in Natural Language Processing (EMNLP)*, Prague, Czech Republic, pages 51–60.

Alexander Fraser and Daniel Marcu. 2007. Measuring Word Alignment Quality for Statistical Machine Translation. In *Computational Linguistics* Vol. 33(3), pages 293–303.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer 2006. Scalable Inference and Training of Context-Rich Syntactic Translation Models. In *Proceedings of the 44th Annual Meeting on Association for Computational Linguistics*, Sydney, Australia, pages 961–968.

Ulf Hermjakob, Kevin Knight, and Hal Daumé III 2008. Name Translation in Statistical Machine Translation: Learning When to Transliterate. In *Proceedings of the 46th Annual Meeting on Association for Computational Linguistics*, Columbus, Ohio, pages 389–397.

Abraham Ittycheriah and Salim Roukos. 2005. A Maximum Entropy Word Aligner for Arabic-English Machine Translation. In *Proceedings of Joint Conference of Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*, Vancouver, British Columbia, Canada, pages 89–96.

Alon Lavie, Alok Parlikar and Vamshi Ambati. 2008. Syntax-Driven Learning of Sub-Sentential Translation Equivalents and Translation Rules from Parsed Parallel Corpora. In *Proceedings of the ACL/HLT Second Workshop on Syntax and Structure in Statistical Translation (SSST-2)*, Columbus, Ohio, pages 87–95.

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

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. In *Computational Linguistics* Vol. 29(1), pages 19–51.

Franz Josef Och and Hermann Ney. 2004. The Alignment Template Approach to Statistical Machine Translation. In *Computational Linguistics* Vol. 30(4), pages 417–449.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Philadelphia, PA, pages 311–318.

Radu Soricut and Daniel Marcu. 2003. Sentence Level Discourse Parsing Using Syntactic and Lexical Information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, Edmonton, Canada, pages 149–156.

Andy Way, Nano Gough. 2003. wEBMT: developing and validating an example-based machine translation system using the world wide web In *Computational Linguistics* Vol. 29(3), pages 421–457.

Ventsislav Zhechev, Andy Way. 2008. Automatic Generation of Parallel Treebanks. In *Proceedings of 22nd International Conference on Computational Linguistics (COLING)*, Manchester, UK, pages 1105–1112.

Entity Extraction via Ensemble Semantics

Marco Pennacchiotti
Yahoo! Labs
Sunnyvale, CA, 94089
pennac@yahoo-inc.com

Patrick Pantel
Yahoo! Labs
Sunnyvale, CA, 94089
ppantel@yahoo-inc.com

Abstract

Combining information extraction systems yields significantly higher quality resources than each system in isolation. In this paper, we generalize such a mixing of sources and features in a framework called *Ensemble Semantics*. We show very large gains in entity extraction by combining state-of-the-art distributional and pattern-based systems with a large set of features from a webcrawl, query logs, and Wikipedia. Experimental results on a web-scale extraction of actors, athletes and musicians show significantly higher mean average precision scores (29% gain) compared with the current state of the art.

1 Introduction

Mounting evidence shows that combining information sources and information extraction algorithms leads to improvements in several tasks such as fact extraction (Paşca et al., 2006), open-domain IE (Talukdar et al., 2008), and entailment rule acquisition (Mirkin et al., 2006). In this paper, we show large gains in entity extraction by combining state-of-the-art distributional and pattern-based systems with a large set of features from a 600 million document webcrawl, one year of query logs, and a snapshot of Wikipedia. Further, we generalize such a mixing of sources and features in a framework called *Ensemble Semantics*.

Distributional and pattern-based extraction algorithms capture aspects of paradigmatic and syntagmatic dimensions of semantics, respectively, and are believed to be quite complementary. Paşca et al. (2006) showed that filtering facts, extracted by a pattern-based system, according to their arguments' distributional similarity with seed facts yielded large precision gains. Mirkin et al. (2006) showed similar gains on the task of acquiring lexical entailment rules by exploring a supervised combination of distributional and pattern-based algorithms using an ML-based SVM classifier.

This paper builds on the above work, by studying the impact of various sources of features external to distributional and pattern-based algorithms, on the task of entity extraction. Mirkin et al.'s results are corroborated on this task and large and significant gains over this baseline are obtained by incorporating 402 features from a webcrawl, query logs and Wikipedia. We extracted candidate entities for the classes *Actors*, *Athletes* and *Musicians* from a webcrawl using a variant of Paşca et al.'s (2006) pattern-based engine and Pantel et al.'s (2009) distributional extraction system. A gradient boosted decision tree is used to learn a regression function over the feature space for ranking the candidate entities. Experimental results show 29% gains (19% nominal) in mean average precision over Mirkin et al.'s method and 34% gains (22% nominal) in mean average precision over an unsupervised baseline similar to Paşca et al.'s method. Below we summarize the contributions of this paper:

- We explore the hypothesis that although distributional and pattern-based algorithms are complementary, they do not exhaust the semantic space; other sources of evidence can be leveraged to better combine them;
- We model the mixing of knowledge sources and features in a novel and general information extraction framework called *Ensemble Semantics*; and
- Experiments over an entity extraction task show that our model achieves large and significant gains over state-of-the-art extractors. A detailed analysis of feature correlations and interactions shows that query log and webcrawl features yield the highest gains, but easily accessible Wikipedia features also improve over current state-of-the-art systems.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 238–247,
Singapore, 6-7 August 2009.

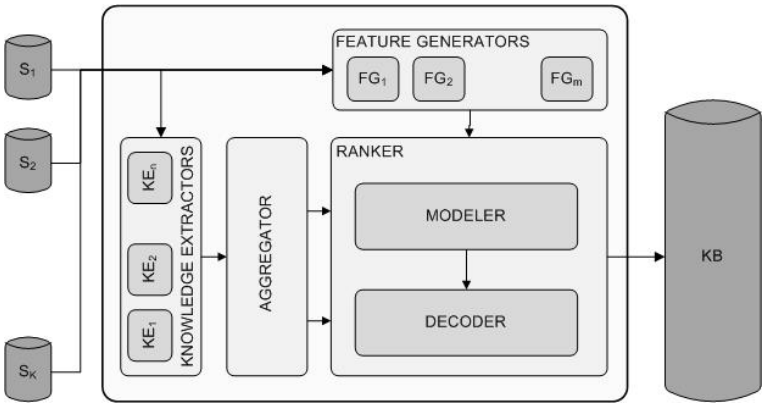


Figure 1: The *Ensemble Semantics* framework for information extraction.

The remainder of this paper is organized as follows. In the next section, we present our Ensemble Semantics framework and outline how various information extraction systems can be cast into the framework. Section 3 then presents our entity extraction system as an instance of Ensemble Semantics, comparing and contrasting it with previous information extraction systems. Our experimental methodology and analysis is described in Section 4 and shows empirical evidence that our extractor significantly outperforms prior art. Finally, Section 5 concludes with a discussion and future work.

2 Ensemble Semantics

Ensemble Semantics (ES) is a general framework for modeling information extraction algorithms that combine multiple sources of information and multiple extractors. The ES framework allows to:

- Represent multiple sources of knowledge and multiple extractors of that knowledge;
- Represent multiple sources of features;
- Integrate both rule-based and ML-based knowledge ranking algorithms; and
- Model previous information extraction systems (i.e., backwards compatibility).

2.1 ES Framework

ES can be instantiated to extract various types of knowledge such as entities, facts, and lexical entailment rules. It can also be used to better understand the commonalities and differences between existing information extraction systems.

After presenting the framework in the next section, Section 2.2 shows how previous information extraction algorithms can be cast into ES. In Section 3 we describe our novel entity extraction algorithm based on ES.

The ES framework is illustrated in Figure 1. It decomposes the process of information extraction into the following components:

Sources (S): textual repositories of information, either structured (e.g., a database such as DBpedia), semi-structured (e.g., Wikipedia Infoboxes or HTML tables) or unstructured (e.g., news articles or a webcrawl).

Knowledge Extractors (KE): algorithms responsible for extracting candidate instances such as entities or facts. Examples include fact extraction systems such as (Cafarella et al., 2005) and entity extraction systems such as (Paşca, 2007).

Feature Generators (FG): methods that extract evidence (features) of knowledge in order to decide which candidate instances extracted from KEs are correct. Examples include capitalization features for named entity extractors, and the distributional similarity matrix used in (Paşca et al., 2006) for filtering facts.

Aggregator (A). A module collecting and assembling the instances coming from the different extractors. This module keeps the *footprint* of each instance, i.e. the number and the type of the KEs that extracted the instance. This information can be used by the Ranker module to build a ranking strategy, as described below.

Ranker (R): a module for ranking the knowledge instances returned from KEs using the features generated by FGs. Ranking algorithms may be rule-based (e.g., the one using a threshold on distributional similarity in (Paşca et al., 2006)) or ML-based (e.g., the SVM model in (Mirkin et al., 2006) for combining pattern-based and distributional features).

The Ranker is composed of two sub-modules: the Modeler and the Decoder. The *Modeler* is responsible for creating the model which ranks the candidate instances. The *Decoder* collects the candidate instances from the Aggregator, and applies the model to produce the final ranking.

In rule-based systems, the Modeler corresponds to a set of manually crafted or automatically induced rules operating on the features (e.g. a combination of thresholds). In ML-based systems, it is an actual machine learning algorithm, that takes as input a set of labeled training instances, and builds the model according to their features. Training instances can be obtained as a subset of those collected by the Aggregator, or from some external resource. In many cases, training instances are manually labeled by human experts, through a long and costly editorial process.

Information sources (S) serve as inputs to the system. Some sources will serve as sources for knowledge extractors to generate candidate instances, some will serve as sources for feature generators to generate features or evidence of knowledge, and some will serve as both.

2.2 Related Work

To date, most information extraction systems rely on a model composed of a single source S, a single extractor KE and a single feature generator FG. For example, many classic relation extraction systems (Hearst, 1992; Riloff and Jones, 1999; Pantel and Pennacchiotti, 2006; Paşca et al., 2006) are based on a single pattern-based extractor KE, which is seeded with a set of patterns or instances for a given relation (e.g. the pattern *'X starred in Y'* for the *act-in* relation). The extractor then iteratively extracts new instances until a stop condition is met. The resulting extractor scores are proposed by FG as a feature. The *Ranker* simply consists of a sorting function on the feature from FG.

Systems such as the above that do not consist of multiple sources, knowledge extractors or feature generators are not considered Ensemble Semantics models, even though they can be cast into the framework. Recently, some researchers have explored more complex systems, having multiple sources, extractors and feature generators. Below we show examples and describe how they map as Ensemble Semantics systems. We use this characterization to clearly outline how our proposed entity extraction system, proposed in Section 3, differs from previous work.

Talukdar et al. (2008) present a weakly-supervised system for extracting large sets of class-instance pairs using two knowledge extractors: a pattern-based extractor supported by distributional evidence, which harvests candidate pairs from a Web corpus; and a table extractor that harvests candidates from Web tables. The *Ranker* uses graph random walks to combine the information of the two extractors and output the final list. The authors show large improvements in coverage with little precision loss.

Mirkin et al. (2006) introduce a machine learning system for extracting lists of lexical entailments (e.g. 'government' → 'organization'). They rely on two knowledge extractors, operating on a same large textual source: a pattern-based extractor, leveraging the Hearst (1992) *is-a* patterns; and a distributional extractor applied to a set of entailment seeds. Candidate instances are passed to an SVM *Ranker*, which uses features stemming from the two extractors, to decide which instances are output in the final list. The authors report a +9% increase in F-measure over a rule-based system that takes the union of the instances extracted by the two modules.

Other examples include the system for taxonomic-relation extraction by Cimiano et al. (2005), using a pool of feature generators based on pattern-based, distributional and WordNet techniques; and Paşca and Van Durme's (2008) system that uses a Web corpus and query logs to extract semantic classes and their attributes.

Similarly to these methods, our proposed entity extractor (Section 3) utilizes multiple sources and extractors. A key difference of our method lies in the Feature Generator module. We propose several generators resulting in 402 features extracted from Web pages, query logs and Wikipedia articles. The use of these features results in dramatic performance improvements, reported in Section 4.

3 ES for Entity Extraction

Entity extraction is a fundamental task in NLP responsible for extracting instances of semantic classes (e.g., *'Brad Pitt'* and *'Tom Hanks'* are instances of the class *Actors*). It forms a building block for various NLP tasks such as ontology learning (Cimiano and Staab, 2004) and co-reference resolution (Mc Carthy and Lehn-

Family	*Type*		*Features*
Web (w)	Frequency	(wF)	term frequency; document frequency; term frequency as noun phrase
	Pattern	(wP)	confidence score returned by KE_{pat}; pmi with the 100 most reliable patterns used by KE_{pat}
	Distributional	(wD)	distributional similarity with the centroid in KE_{dis}; distributional similarities with each seed in $\mathcal{S}$
	Termness	(wT)	ratio between term frequency as noun phrase and term frequency; pmi between internal tokens of the instance; capitalization ratio
Query log (q)	Frequency	(qF)	number of queries matching the instance; number of queries containing the instance
	Co-occurrence	(qC)	query log pmi with any seed in $\mathcal{S}$
	Pattern	(qP)	pmi with a set of trigger words $\mathcal{T}$ (i.e., the 10 words in the query logs with highest pmi with $\mathcal{S}$)
	Distributional	(qD)	distributional similarity with $\mathcal{S}$ (vector coordinates consist of the instance's pmi with the words in $\mathcal{T}$)
	Termness	(qT)	ratio between the two frequency features F
Web table (t)	Frequency	(tF)	table frequency
	Co-occurrence	(tC)	table pmi with $\mathcal{S}$; table pmi with any seed in $\mathcal{S}$
Wikipedia (k)	Frequency	(kF)	term frequency
	Co-occurrence	(kC)	pmi with any seed in $\mathcal{S}$
	Distributional	(kD)	distributional similarity with $\mathcal{S}$

Table 1: Feature space describing each candidate instance ($\mathcal{S}$ indicates the set of seeds for a given class).

ert, 2005). Search engines such as Yahoo, Live, and Google collect large sets of entities (Paşca, 2007; Chaudhuri et al., 2009) to better interpret queries (Tan and Peng, 2006), to improve query suggestions (Cao et al., 2008) and to understand query intents (Hu et al., 2009). Entity extraction differs from the similar task of named entity extraction, in that classes are more fine-grained and possibly overlapping.

Below, we propose a new method for entity extraction built on the ES framework (Section 3.1). Then, we comment on related work in entity extraction (Section 3.2).

3.1 ES Entity Extraction Model

In this section, we propose a novel entity extraction model following the Ensemble Semantics framework presented in Section 2. The sources of our systems can come from any textual corpus. In our experiments (described in Section 4.1), we extracted entities from a large crawl of the Web, and generated features from this crawl as well as query logs and Wikipedia.

3.1.1 Knowledge extractors

Our system relies on two knowledge extractors: one pattern-based and the other distributional.

Pattern-based extractor (KE_{pat}). We reimplemented Paşca et al.'s (2006) state-of-the-art web-scale fact extractor, which, given seed instances of a binary relation, finds instances of that relation. We extract entities of a class, such as *Actors*, by instantiating typical relations involving that class such as *act-in(Actor, Movie)*. We instantiate such relations instead of the classical *is-a* patterns since these have been shown to bring in too many false positives, see (Pantel and Pennacchiotti, 2006) for a discussion of such generic patterns. The extractor's confidence score for each instance is used by the *Ranker* to score the entities being extracted. Section 4.1 lists the system parameters we used in our experiments.

Distributional extractor (KE_{dis}). We use Pantel et al.'s (2009) distributional entity extractor. For each noun in our source corpus, we build a context vector consisting of the noun chunks preceding and following the target noun, scored using pointwise mutual information (pmi). Given a small set of seed entities $\mathcal{S}$ of a class, the extractor computes the centroid of the seeds' context vectors as a geometric mean, and then returns all nouns whose similarity with the centroid exceeds a threshold τ (using the cosine measure between the context vectors). Full algorithmic details are presented in (Pantel et al., 2009). Section 4.1 lists the threshold and text preprocessing algorithms used in our experiments.

The *Aggregator* simply takes a union of the entities discovered by the two extractors.

3.1.2 Feature generators

Our model includes four feature generators, which compute a total of 402 features (full set described in Table 1). Each generator extracts from a specific source a *feature family*, as follows:

- *Web (w)*: a body of 600 million documents

crawled from the Web at Yahoo! in 2008;

- *Query logs (q)*: one year of web search queries issued to the Yahoo! search engine;
- *Web tables*: all HTML inner tables extracted from the above *Web* source; and
- *Wikipedia*: an official Wikipedia dump from February 2008, consisting of about 2 million articles.

Feature families are further subclassified into five types: *frequency (F)* (frequency-based features); *co-occurrence (C)* (features capturing first order co-occurrences between an instance and class seeds); *distributional (D)* (features based on the distributional similarity between an instance and class seeds); *pattern (P)* (features indicating class-specific lexical pattern matches); and *termness (T)* (features used to distinguish well-formed terms such as *'Brad Pitt'* from ill-formed ones such as *'with Brad Pitt'*). The seeds $\mathcal{S}$ used in many of the feature families are the same seeds used by the KE_{pat} extractor, described in Section 3.1.1.

The different seed families are designed to capture different semantic aspects: paradigmatic (D), syntagmatic (C and P), popularity (F), and term cohesiveness (T).

3.1.3 ML-based Ranker

Our *Modeler* adopts a supervised ML regression model. Specifically, we use a Gradient Boosted Decision Tree regression model - GBDT (Friedman, 2001), which consists of an ensemble of decision trees, fitted in a forward step-wise manner to current residuals. Friedman (2001) shows that by drastically easing the problem of overfitting on training data (which is common in boosting algorithms), GDBT competes with state-of-the-art machine learning algorithms such as SVM (Friedman, 2006) with much smaller resulting models and faster decoding time. The model is trained on a manually annotated random sample of entities taken from the *Aggregator*, using the features generated by the four generators presented in Section 3.1.2. The *Decoder* then ranks each entity according to the trained model.

3.2 Related Work

Entity extraction systems follow two main approaches: pattern-based and distributional. The *pattern-based approach* leverages lexico-syntactic patterns to extract instances of a given class. Most commonly used are *is-a* pattern families such as those first proposed by Hearst (1992) (e.g., *'Y such as X'* for matching *'actors such as Brad Pitt'*). Minimal supervision is used in the form of small sets of manually provided seed patterns or seed instances. This approach is very common in both the NLP and Semantic Web communities (Cimiano and Staab, 2004; Cafarella et al., 2005; Pantel and Pennacchiotti, 2006; Paşca et al., 2006).

The *distributional approach* uses contextual evidence to model the instances of a given class, following the distributional hypothesis (Harris, 1964). Weakly supervised, these methods take a small set of seed instances (or the class label) and extract new instances from noun phrases that are most similar to the seeds (i.e., that share similar contexts). Following Lin (1998), example systems include Fleischman and Hovy (2002), Cimiano and Volker (2005), Tanev and Magnini (2006), and Pantel et al. (2009).

4 Experimental Evaluation

This section reports our experiments, showing the effectiveness of our entity extraction system and the importance of our different feature families.

4.1 Experimental Setup

Evaluated classes. We evaluate our system over three classes: *Actors* (movie, tv and stage actors); *Athletes* (professional and amateur); *Musicians* (singers, musicians, composers, bands, and orchestras)

System setup. We instantiated our knowledge extractors, KE_{pat} and KE_{dis} from Section 3.1.1, over our *Web* crawl of 600 million documents (see Section 3.1.2). The documents were preprocessed using Brill's POS-tagger (Brill, 1995) and the Abney's chunker (Abney, 1991). For $\boldsymbol{KE_{dis}}$, context vectors are extracted for noun phrases recognized as NP-chunks with removed modifiers. The vector space includes the 250M most frequent noun chunks in the corpus. KE_{dis} returns as instances all noun phrases having a similarity with the seeds' centroid above $\tau = 0.005$[1]. The sets of seeds $\mathcal{S}$ for KE_{dis} include 10, 24 and 10 manually chosen instances for respectively the *Actors*, *Athletes* and *Musicians* classes[2]. The sets of seeds $\mathcal{P}$ for KE_{pat}

[1] Experimentally set on an independent development set.

[2] The higher number of seeds for *Athletes* is chosen to cover different sports.

Dataset	*Actors*	*Athletes*	*Musicians*
KE_{pat}	58,005	40,816	125,657
KE_{dis}	72,659	24,380	24,593
$KE_{pat} \cup KE_{dis}$	113,245	61,709	142,694
$KE_{pat} \cap KE_{dis}$	17,419	3,487	7,556
$\mathcal{R}$	500	500	500
	P=80	*P=258*	*P=134*
	N=420	*N=242*	*N=366*

Table 2: Number of extracted instances and the sample sizes (P and N indicate positive and negative annotations).

include 11, 8 and 9 pairs respectively for the *Actors* (relation *acts-in*), *Athletes* (relation *plays-for*) and *Musicians* (relation *part-of-band*) classes. Table 6 lists all seeds for both KE_{dis} and KE_{pat}. The ***GBDT ranker*** uses an ensemble of 300 trees.[3]

Goldset Preparation. The number of instances extracted by KE_{pat} and KE_{dis} for each class is reported in Table 2. For each class, we extract a random sample $\mathcal{R}$ of 500 instances from $KE_{pat} \cup KE_{dis}$. A pool of 10 paid expert editors annotated the instances of each class in $\mathcal{R}$ as positive or negative. Inter-annotator overlap was 0.88. Uncertain instances were manually adjudicated by a separate paid expert editor, yielding a gold standard dataset for each class.

Evaluation Metrics. Entity extraction performance is evaluated using the *average precision* (AP) statistic, a standard information retrieval measure for evaluating ranking algorithms, defined as:

$$AP(L) = \frac{\sum_{i=1}^{|L|} P(i) \cdot corr(i)}{\sum_{i=1}^{|L|} corr(i)} \qquad (1)$$

where L is a ranked list produced by an extractor, $P(i)$ is the precision of L at rank i, and *corr(i)* is 1 if the instance at rank i is correct, and 0 otherwise. AP is computed over $\mathcal{R}$ for each class.

We also evaluate the *coverage*, i.e. the percentage of instances extracted by a system wrt those extracted by all systems.

In order to accurately compute statistical significance, our experiments are performed using 10-fold cross validation.

Baselines and comparisons. We compare our proposed ES entity extractor, using different feature configurations, with state-of-the-art systems (referred to as baselines *B** below):

[3]GBDT model parameters were experimentally set on an independent development set as follows: trees=300, shrinkage=0.01, max_nodes_per_tree=12, sample_rate=0.5.

System	Actors		Athletes		Musicians	
	AP	*Cov*	*AP*	*Cov*	*AP*	*Cov*
B1	0.729	51.2%	0.616	66.1%	0.570	88.1%
B2	0.618	64.1%	0.687	39.5%	0.681	17.2%
B3	0.676	100%	0.664	100%	0.576	100%
B4	0.715	100%	0.697	100%	0.579	100%
ES-all	**0.860**‡	100%	**0.915**‡	100%	**0.788**‡	100%

Table 3: Average precision (AP) and coverage (Cov) results for our proposed system *ES-all* and the baselines. ‡ indicates AP statistical significance at the 0.95 level wrt all baselines.

ES-all. Our ES system, using KE_{pat} and KE_{dis}, the full set of feature families described in Section 3.1.2, and the GBDT ranker.

B1. KE_{pat} alone, a state-of-the-art pattern-based extractor reimplementing (Paşca et al., 2006), where the *Ranker* assigns scores to instances using the confidence score returned by KE_{pat}.

B2. KE_{dis} alone, a state-of-the-art distributional system implementing (Pantel et al., 2009), where the *Ranker* assigns scores to instances using the similarity score returned by KE_{dis} alone.

B3. A rule-based ES system, combining *B1* and *B2*. This system uses both KE_{pat} and KE_{dis} as extractors, and a *Ranker* that assigns scores to instances according to the sum of their normalized confidence scores.

B4. A state-of-the-art machine learning system based on (Mirkin et al., 2006). This ES system uses KE_{pat} and KE_{dis} as extractors. The *Ranker* is a GBDT regression model, using the full sets of features derived from the two extractors, i.e., wP and wD (see Table 1). GBDT parameters are set as for our proposed *ES-all* system.

4.2 Experimental Results

Table 3 summarizes the average-precision (AP) and coverage results for our *ES-all* system and the baselines. Figure 2 reports the precision at each rank for the *Athletes* class (the other two classes follow similar trends). Table 6 lists the top-10 entities discovered for each class on one test fold. *ES-all* outperforms all baselines in AP (all results are statistically significant at the 0.95 level), offering at the same time full coverage[4].

[4]Recall that coverage is reported relative to all instances retrieved by extractors KE_{pat} and KE_{dis}.

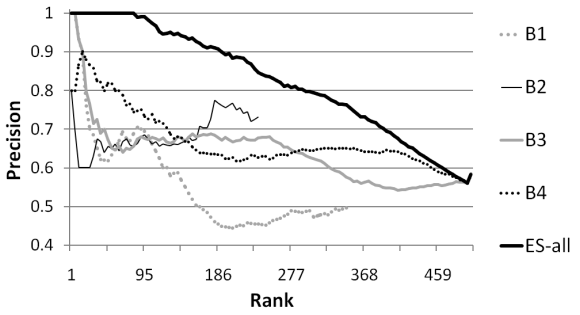


Figure 2: Precision at rank for the different systems on the *Athletes* class.

System	*AP*			*MAP*
	Actors	**Athletes**	**Musicians**	
B3	0.676	0.664	0.576	0.639
B4	0.715	0.697	0.579	0.664
B4+w	$0.813^{\ddagger}$	$0.908^{\ddagger}$	$0.724^{\ddagger}$	$0.815^{\ddagger}$
B4+q	$0.815^{\ddagger}$	$0.905^{\ddagger}$	$0.743^{\ddagger}$	$0.821^{\ddagger}$
B4+t	$0.784^{\dagger}$	$0.825^{\ddagger}$	$0.727^{\ddagger}$	$0.779^{\ddagger}$
B4+k	$0.776^{\dagger}$	$0.825^{\ddagger}$	0.624	$0.741^{\dagger}$
B4+w+q	$0.835^{\ddagger}$	$0.915^{\ddagger}$	$0.758^{\ddagger}$	$0.836^{\ddagger}$
B4+w+t	$0.840^{\ddagger}$	$0.906^{\ddagger}$	$0.774^{\ddagger}$	$0.840^{\ddagger}$
B4+w+k	$0.814^{\ddagger}$	$0.903^{\ddagger}$	$0.725^{\ddagger}$	$0.814^{\ddagger}$
B4+q+t	$0.847^{\ddagger}$	$0.910^{\ddagger}$	$0.774^{\ddagger}$	$0.844^{\ddagger}$
B4+q+k	$0.832^{\ddagger}$	$0.906^{\ddagger}$	$0.748^{\ddagger}$	$0.829^{\ddagger}$
B4+t+k	$0.817^{\ddagger}$	$0.861^{\ddagger}$	$0.743^{\ddagger}$	$0.807^{\ddagger}$
B4+w+q+t	$0.846^{\ddagger}$	$\mathbf{0.917}^{\ddagger}$	$0.782^{\ddagger}$	$0.849^{\ddagger}$
B4+w+q+k	$0.841^{\ddagger}$	$0.916^{\ddagger}$	$0.756^{\ddagger}$	$0.838^{\ddagger}$
B4+w+t+k	$0.835^{\ddagger}$	$0.906^{\ddagger}$	$0.783^{\ddagger}$	$0.841^{\ddagger}$
Es-all	$\mathbf{0.860}^{\ddagger}$	$0.915^{\ddagger}$	$\mathbf{0.788}^{\ddagger}$	$\mathbf{0.854}^{\ddagger}$

Table 4: Overall AP results of the different feature configurations, compared to two baselines. † indicates statistical significance at the 0.95 level wrt *B3*. ‡ indicates statistical significance at 0.95 level wrt both *B3* and *B4*.

Our simple rule-based combination baseline, *B3*, leads to a large increase in coverage wrt the individual extractors alone (*B1* and *B2*) without significant impact on precision. The supervised ML-based combination baseline (*B4*) consistently improves AP across classes wrt the rule-based combination (*B3*), but without statistical significance. These results corroborate those found in (Mirkin et al., 2006), where this ML-based combination was reported to be significantly better than a rule-based one on the task of lexical entailment acquisition.

The large set of features adopted in *ES-all* accounts for a dramatic improvement in AP, indicating that existing state-of-the-art systems for entity extraction (reflected by our baselines strategies) are not making use of enough semantic cues. The adoption of evidence other than distributional and pattern-based, such as features coming from web documents, HTML tables and query logs, is here demonstrated to be highly valuable.

The above empirical claim can be grounded and corroborated by a deeper semantic analysis. From a semantic perspective, the above results translate in the observation that distributional and pattern-based evidence do not completely capture all semantic aspects of entities. Other evidence, such as popularity, term cohesiveness and co-occurrences capture other aspects. For instance, in one of our *Actors* folds, the *B3* system ranks the incorrect instance *'Tom Sellek'* (a misspelling of *'Tom Selleck'*) in 9^{th} position (out of 142), while *ES-all* lowers it to the 33^{rd} position, by relying on table-based features (intuitively, tables contain much fewer misspelling than running text). Other than misspellings, *ES-all* fixes errors that are either typical of distributional approaches, such as the inclusion of instances of other classes (e.g. the movie *'Someone Like You'* often appears in contexts similar to those of *actors*); errors typical of pattern-based approaches, such as incorrect instances highly-associated with an ambiguous pattern (e.g., the pattern *'X of the rock band Y'* for finding *Musicians* matched an incorrect instance *'song submission'*); or errors typical of both, such as the inclusion of common nouns (e.g. *'country music hall'*) or too generic last names (e.g. *'Johnson'*). *ES-all* successfully recovers all these error by using termness, co-occurrence and frequency features.

We also compare *ES-all* with a state-of-the-art random walk system (*RW*) presented by Talukdar et al. (2008) (see Section 2.2 for a description). As we could not reimplement the system, we report the following indirect comparison. *RW* was evaluated on five entity classes, one of which, *NFL players*, overlaps with our *Athletes* class. On this class, they report 0.95 precision on the top-100 ranked entities. Unfortunately, they do not report coverage or recall statistics, making the interpretation of this analysis difficult. In an attempt to compare *RW* with *ES-all*, we evaluated the precision of our top-100 *Athletes*, obtaining 0.99. Using a random sample of our extracted *Athletes*, we approximate the precision of the top-22,000 *Athletes* to be 0.97 ± 0.01 (at the 0.95 level).

4.3 Feature Analysis

Feature family analysis: Table 4 reports the average precision (AP) for our system using different feature family combinations (see Table 1). Column 1 reports the family combinations; columns

2-4 report AP for each class; and column 5 reports the mean-average-precision (*MAP*) across classes. In all configurations, except the k family alone, and along all classes, our system significantly outperforms (at the 0.95 level) the baselines.

Rows 3-6 report the performance of each feature family alone. w and t are consistently better than q and k, across all classes. k is shown to be the least useful family. This is mostly due to data sparseness, e.g., in our experiments almost 40% of the test instances in the *Actors* sample do not have any occurrence in Wikipedia. However, without access to richer resources such as a webcrawl or query logs, the features from k do indeed provide large gains over current baselines (on average +10.2% and +7.7% over *B3* and *B4*).

Rows 7-12 report results for combinations of two feature families. All combinations (except those with k) appear valuable, substantially increasing the single-family results in rows 3-6, indicating that combining different feature families (as suggested by the ES paradigm) is helpful. Second, it indicates that q, w and t convey complementary information, thus boosting the regression model when combined together. It is interesting to notice that *q+t* tends to be the best combination, surprising given that t alone did not show high performance (row 5). One would expect the combination *q+w* to outperform *q+t*, but the good performance of *q+t* is mainly due to the fact that these two families are more complementary than *q+w*. To verify this intuition, we compute the Spearman correlation coefficient r among the rankings produced by the different combinations. As expected, *q* and *w* have a higher correlation ($r = 0.82$) than *q* and *t* ($r = 0.67$) and *w* and *t* ($r = 0.66$), suggesting that q and w tend to subsume each other (i.e. no added information for the regression model).

Rows 13-15 report results for combinations of three feature families. As expected, the best combination is *q+w+t* with an average precision nearly identical to the full *ES-all* system. If one has access to Web or query log sources, then the value of the Wikipedia features tends to be subsumed by our web and query log features.

Feature by feature analysis: The feature families analyzed in the previous section consist of 402 features. For each trained GBDT model, one can inspect the resulting most important features (Friedman, 2001). Consistently, the two most important features for *ES-all* are, as expected, the confidence scores of KE_{pat} and KE_{dis}. This suggests that syntagmatic and paradigmatic information are most important in defining the semantics of entities. Also very important, in third position, is a feature from qT, namely the ratio between the number of queries matching the instance and the number of queries containing it as a substring. This feature is a strong indicator of termness.

System	*AP*			*MAP*
	Actors	**Athletes**	**Musicians**	
B4	0.715	0.697	0.579	0.664
B4+**w**	**0.813**	**0.908**	**0.724**	**0.815**
B4+wF	0.798	0.865	0.679	0.781
B4+wT	0.806	0.891	0.717	0.805
B4+**t**	**0.784**	**0.825**	**0.727**	**0.779**
B4+tF	0.760	0.802	0.701	0.781
B4+tC	0.771	0.815	0.718	0.805
B4+**q**	**0.815**	**0.905**	**0.743**	**0.821**
B4+qF	0.786	0.890	0.693	0.790
B4+qC	0.715	0.738	0.581	0.678
B4+qD	0.735	0.709	0.644	0.696
B4+qP	0.779	0.796	0.648	0.741
B4+qT	0.780	0.868	0.725	0.791
B4+qF+qW+qT	0.816	0.906	0.743	0.822
ES-all	**0.860**	**0.915**	**0.788**	**0.854**

Table 5: Ablation study of the web (w), query-log (q) and table (t) features (bold letters indicate whole feature families).

Webcrawl term frequencies and document frequencies (from the wF set) are also important. Very frequent and infrequent instances were found to be often incorrect (e.g., respectively *'song'* and *'Brad Pitttt'*). Table PMI (a feature in the qC family) also ranked high in importance: instances that co-occurr very frequently in the same column/row with seeds $\mathcal{S}$ are often found to be correct (e.g., a column containing the seeds *'Brad Pitt'* and *'Tom Hanks'* will likely contains other actors). Other termness (T), frequency-based (F) and co-occurrence (C) features also play some role in the model.

Variable importance is only an intrinsic indicator of feature relevance. In order to better assess the actual impact of the single features on AP, we ran our system on each feature type: results for the web (w), query log (q) and table (t) families are reported in Table 5. For reason of space constraints, we here only focus on some high level observations. The set of web termness features (wT) and frequency features (wF) are alone able to provide a large improvement over *B4* (row 1), while their combination (row 2) does not improve much over the features taken individually.

Seed instances for KE_{dis}

Actors	Athletes			Musicians
Jodie Foster	Bob Gibson	Jared Allen	Randy Moss	Rise Against the Machine
Humphrey Bogart	Don Drysdale	Andres Romero	Peyton Manning	Pink Floyd
Anthony Hopkins	Albert Pujols	Kenny Perry	Jerry Rice	Spice Girls
Katharine Hepburn	Yogi Berra	Martin Kaymer	Robert Karlsson	Pussycat Dolls
Christopher Walken	Dejan Bodiroga	Alexander Ovechkin	Gheorghe Hagi	The Beatles
Gene Hackman	Allen Iverson	Shea Weber	Marco Van Basten	Iron Maiden
Diane Keaton	Yao Ming	Patrick Roy	Zinedine Zidane	John Lennon
Edward Norton	Tim Duncan	Alexei Kovalev	Roberto Baggio	Frank Sinatra
Robert Duvall				Led Zeppelin
Hilary Swank				Freddie Mercury

Seed instances for KE_{pat}

Actors	Athletes	Musicians
Dennis Hopper - The Good Life	Dallas cowboys - Julius Crosslin	Kevin Brown - Corndaddy
Tom Hanks - The Terminal	New york Giants - Plaxico Burress	Barry Gibb - The Bee Gees
Julia Roberts - Mona Lisa Smile	Philadelphia Eagles - Danny Amendola	Patty Smyth - Scandal
Kevin Bacon - Footloose	Washington Redskins - Rock Cartwright	Dave Matthews - Dave Mathews Band
Keanu Reeves - The Lake House	New England Patriots - Laurence Maroney	Gwen Stefani - No Doubt
Marlon Brando - Don Jaun Demarco	Buffalo Bills - Xavier Omon	George Michael - Wham
Morgan Freeman - The Shawshank Redemption	Miami Dolphins - Ernest Wilford	Mark Knopfler - Dire Straits
Nicole Kidman - Eyes Wide Shut	New York Jets - Chansi Stuckey	Brian Jones - The Rolling Stones
Al Pacino - The Godfather		Pete Shelley - Buzzcocks
Johnny Depp - Chocolat		
Halle Berry - Monster's Ball		

10-best ranked instances in one test fold

Actors		Athletes		Musicians	
Gordon Tootoosis	Ron Randell	Rumeal Robinson	Todd Warriner	Colin Newman	Wu-tang Clan
Rosalind Chao	Alimi Ballard	Jeff Mcinnis	Hong-chih Kuo	Ghost Circus	Tristan Prettyman
John Hawkes	Fernando Lamas	Ahmad Nivins	Leon Clarke	Ray Dorset	Top Cats
Jeffrey Dean Morgan	Bruno Cremer	Carlos Marchena	Josh Dollard	Plastic Tree	*Roseanne
George Macready	Muhammad Bakri	Chad Kreuter	Robbie Alomar	*Doomwatch	John Moen

Table 6: Listing of all seeds used for KE_{dis} and KE_{pat}, as well as the top-10 entities discovered by *ES-all* on one of our test folds.

This suggests that wT and wF capture very similar information: they are indeed highly correlated ($r = 0.80$). Rows 5-7 refer to web table features: the features tC outperform and subsume the frequency features tF ($r = 0.92$). For query log features (rows 8-14), only qF, qP and qT significantly increase performance. Distributional and co-occurrence features (qD and qC) have very low effect, as they are mostly subsumed by the others. The combination of qF, qP and qT (row 14) performs as well as the whole ***q*** (row 8).

Experiment conclusions: From our experiments, we can draw the following conclusions:

1. Wikipedia features taken alone outperform the baselines, however, web and query log features, if available, subsume Wikipedia features;

2. q, t and w are all important, and should be used in combination, as they drive mostly independent information;

3. the syntagmatic and paradigmatic information conveyed by the two extractors are most relevant, and can be significantly boosted by adding frequency- and termness-based features from other sources.

5 Conclusions and Future Work

In this paper, we presented a general information extraction framework, called Ensemble Semantics, for combining multiple sources of knowledge, and we instantiated the framework to build a novel ML-based entity extraction system. The system significantly outperforms state-of-the-art ones by up to 22% in mean average precision. We provided an in-depth analysis of the impact of our proposed 402 features, their feature families (Web documents, HTML tables, query logs, and Wikipedia), and feature types.

There is ample directions for future work. On entity extraction, exploring more knowledge extractors from different sources (such as the HTML tables and query log sources used for our features) is promising. Other feature types may potentially capture other aspects of the semantics of entities, such as WordNet and search engine click logs. For the ranking system, semi- or weakly-supervised algorithms may provide competing performance to our model with reduced manual labor. Finally, there are many opportunities for applying the general Ensemble Semantics framework to other information extraction tasks such as fact extraction and event extraction.

References

Steven Abney. 1991. Learning taxonomic relations from heterogeneous sources of evidence. In *Principle-Based Parsing*. Kluwer Academic Publishers.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4).

Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. 2005. KnowItNow: Fast, scalable information extraction from the web. In *Proceedings of EMNLP-2005*.

Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware query suggestion by mining click-through and session data. In *Proceedings of KDD-08*, pages 875–883.

Surajit Chaudhuri, Venkatesh Ganti, and Dong Xin. 2009. Exploiting web search to generate synonyms for entities. In *Proceedings of WWW-09*, pages 151–160.

Philipp Cimiano and Steffen Staab. 2004. Learning by googling. *SIGKDD Explorations*, 6(2):24–34.

Philipp Cimiano and Johanna Volker. 2005. Towards large-scale, open-domain and ontology-based named entity classification. In *Proceedings of RANLP-2005*, pages 166–172.

Philipp Cimiano, Aleksander Pivk, Lars Schmidt-Thieme, and Steffen Staab. 2005. Learning taxonomic relations from heterogeneous sources of evidence. In *Ontology Learning from Text: Methods, Evaluation and Applications*, pages 59–73. IOS Press.

Michael Fleischman and Eduard Hovy. 2002. Classification of named entities. In *Proceedings of COLING 2002*.

Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

Jerome H. Friedman. 2006. Recent advances in predictive (machine) learning. *Journal of Classification*, 23(2):175–197.

Zellig Harris. 1964. Distributional structure. In Jerrold J. Katz and Jerry A. Fodor, editors, *The Philosophy of Linguistics*, New York. Oxford University Press.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING-92*, pages 539–545.

Jian Hu, Gang Wang, Fred Lochovsky, Jian tao Sun, and Zheng Chen. 2009. Understanding user's query intent with Wikipedia. In *Proceedings of WWW-09*, pages 471–480.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL-98*.

Joseph F. Mc Carthy and Wendy G Lehnert. 2005. Using decision trees for coreference resolution. In *Proceedings of IJCAI-1995*, pages 1050–1055.

Shachar Mirkin, Ido Dagan, and Maayan Geffet. 2006. Integrating pattern-based and distributional similarity methods for lexical entailment acquisition. In *Proceedings of ACL/COLING-06*, pages 579–586.

Marius Paşca and Benjamin Van Durme. 2008. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *Proceedings of ACL-08*, pages 19–27.

Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In *Proceedings of AAAI-06*, pages 1400–1405.

Marius Paşca. 2007. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of CIKM-07*, pages 683–690, New York, NY, USA.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: A Bootstrapping Algorithm for Automatically Harvesting Semantic Relations. In *Proceedings of ACL-2006*, Sydney, Australia.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of EMNLP-09*, Singapore.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of AAAI-99*, pages 474–479.

Partha Pratim Talukdar, Joseph Reisinger, Marius Paşca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of EMNLP-08*, pages 582–590.

Bin Tan and Fuchun Peng. 2006. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of WWW-06*, pages 1400–1405.

Hristo Tanev and Bernardo Magnini. 2006. Weakly supervised approaches for ontology population. In *Proceedings of EACL-2006*.

Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora

Daniel Ramage, David Hall, Ramesh Nallapati and Christopher D. Manning
Computer Science Department
Stanford University
{dramage,dlwh,nmramesh,manning}@cs.stanford.edu

Abstract

A significant portion of the world's text is tagged by readers on social bookmarking websites. *Credit attribution* is an inherent problem in these corpora because most pages have multiple tags, but the tags do not always apply with equal specificity across the whole document. Solving the credit attribution problem requires associating each word in a document with the most appropriate tags and vice versa. This paper introduces *Labeled LDA*, a topic model that constrains Latent Dirichlet Allocation by defining a one-to-one correspondence between LDA's latent topics and user tags. This allows Labeled LDA to directly learn word-tag correspondences. We demonstrate Labeled LDA's improved expressiveness over traditional LDA with visualizations of a corpus of tagged web pages from *del.icio.us*. Labeled LDA outperforms SVMs by more than 3 to 1 when extracting tag-specific document snippets. As a multi-label text classifier, our model is competitive with a discriminative baseline on a variety of datasets.

1 Introduction

From news sources such as *Reuters* to modern community web portals like *del.icio.us*, a significant proportion of the world's textual data is labeled with multiple human-provided tags. These collections reflect the fact that documents are often about more than one thing—for example, a news story about a highway transportation bill might naturally be filed under both *transportation* and *politics*, with neither category acting as a clear subset of the other. Similarly, a single web page in *del.icio.us* might well be annotated with tags as diverse as *arts*, *physics*, *alaska*, and *beauty*.

However, not all tags apply with equal specificity across the whole document, opening up new opportunities for information retrieval and corpus analysis on tagged corpora. For instance, users who browse for documents with a particular tag might prefer to see summaries that focus on the portion of the document most relevant to the tag, a task we call *tag-specific snippet extraction*. And when a user browses to a particular document, a tag-augmented user interface might provide overview visualization cues highlighting which portions of the document are more or less relevant to the tag, helping the user quickly access the information they seek.

One simple approach to these challenges can be found in models that explicitly address the *credit attribution* problem by associating individual words in a document with their most appropriate labels. For instance, in our news story about the transportation bill, if the model knew that the word "highway" went with *transportation* and that the word "politicians" went with *politics*, more relevant passages could be extracted for either label. We seek an approach that can automatically learn the posterior distribution of each word in a document conditioned on the document's label set.

One promising approach to the credit attribution problem lies in the machinery of Latent Dirichlet Allocation (LDA) (Blei et al., 2003), a recent model that has gained popularity among theoreticians and practitioners alike as a tool for automatic corpus summarization and visualization. LDA is a completely unsupervised algorithm that models each document as a mixture of topics. The model generates automatic summaries of topics in terms of a discrete probability distribution over words for each topic, and further infers per-document discrete distributions over topics. Most importantly, LDA makes the explicit assumption that each word is generated from one underlying topic.

Although LDA is expressive enough to model

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 248–256,
Singapore, 6-7 August 2009.

multiple topics per document, it is not appropriate for multi-labeled corpora because, as an unsupervised model, it offers no obvious way of incorporating a supervised label set into its learning procedure. In particular, LDA often learns some topics that are hard to interpret, and the model provides no tools for tuning the generated topics to suit an end-use application, even when time and resources exist to provide some document labels.

Several modifications of LDA to incorporate supervision have been proposed in the literature. Two such models, Supervised LDA (Blei and McAuliffe, 2007) and DiscLDA (Lacoste-Julien et al., 2008) are inappropriate for multiply labeled corpora because they limit a document to being associated with only a single label. Supervised LDA posits that a label is generated from each document's empirical topic mixture distribution. DiscLDA associates a single categorical label variable with each document and associates a topic mixture with each label. A third model, MM-LDA (Ramage et al., 2009), is not constrained to one label per document because it models each document as a bag of words with a bag of labels, with topics for each observation drawn from a shared topic distribution. But, like the other models, MM-LDA's learned topics do not correspond directly with the label set. Consequently, these models fall short as a solution to the credit attribution problem. Because labels have meaning to the people that assigned them, a simple solution to the credit attribution problem is to assign a document's words to its labels rather than to a latent and possibly less interpretable semantic space.

This paper presents *Labeled LDA* (L-LDA), a generative model for multiply labeled corpora that marries the multi-label supervision common to modern text datasets with the word-assignment ambiguity resolution of the LDA family of models. In contrast to standard LDA and its existing supervised variants, our model associates each label with one topic in direct correspondence. In the following section, L-LDA is shown to be a natural extension of both LDA (by incorporating supervision) and Multinomial Naive Bayes (by incorporating a mixture model). We demonstrate that L-LDA can go a long way toward solving the credit attribution problem in multiply labeled documents with improved interpretability over LDA (Section 4). We show that L-LDA's credit attribution ability enables it to greatly outperform support vector machines on a tag-driven snippet extraction task on web pages from *del.icio.us* (Section 6). And despite its generative semantics, we show that Labeled LDA is competitive with a strong baseline discriminative classifier on two multi-label text classification tasks (Section 7).

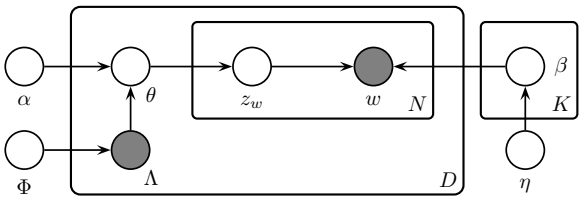


Figure 1: Graphical model of Labeled LDA: unlike standard LDA, both the label set $\mathbf{\Lambda}$ as well as the topic prior $\boldsymbol{\alpha}$ influence the topic mixture $\boldsymbol{\theta}$.

2 Labeled LDA

Labeled LDA is a probabilistic graphical model that describes a process for generating a labeled document collection. Like Latent Dirichlet Allocation, Labeled LDA models each document as a mixture of underlying topics and generates each word from one topic. Unlike LDA, L-LDA incorporates supervision by simply constraining the topic model to use only those topics that correspond to a document's (observed) label set. The model description that follows assumes the reader is familiar with the basic LDA model (Blei et al., 2003).

Let each document d be represented by a tuple consisting of a list of word indices $\mathbf{w}^{(d)} = (w_1, \ldots, w_{N_d})$ and a list of binary topic presence/absence indicators $\mathbf{\Lambda}^{(d)} = (l_1, \ldots, l_K)$ where each $w_i \in \{1, \ldots, V\}$ and each $l_k \in \{0, 1\}$. Here N_d is the document length, V is the vocabulary size and K the total number of unique labels in the corpus.

We set the number of topics in Labeled LDA to be the number of unique labels K in the corpus. The generative process for the algorithm is found in Table 1. Steps 1 and 2—drawing the multinomial topic distributions over vocabulary $\boldsymbol{\beta}_k$ for each topic k, from a Dirichlet prior $\boldsymbol{\eta}$—remain the same as for traditional LDA (see (Blei et al., 2003), page 4). The traditional LDA model then draws a multinomial mixture distribution $\boldsymbol{\theta}^{(d)}$ over all K topics, for each document d, from a Dirichlet prior $\boldsymbol{\alpha}$. However, we would like to restrict $\boldsymbol{\theta}^{(d)}$ to be defined only over the topics that correspond to

1 For each topic $k \in \{1, \ldots, K\}$:
2 Generate $\boldsymbol{\beta}_k = (\beta_{k,1}, \ldots, \beta_{k,V})^T \sim \text{Dir}(\cdot|\boldsymbol{\eta})$
3 For each document d:
4 For each topic $k \in \{1, \ldots, K\}$
5 Generate $\Lambda_k^{(d)} \in \{0, 1\} \sim \text{Bernoulli}(\cdot|\Phi_k)$
6 Generate $\boldsymbol{\alpha}^{(d)} = L^{(d)} \times \boldsymbol{\alpha}$
7 Generate $\boldsymbol{\theta}^{(d)} = (\theta_{l_1}, \ldots, \theta_{l_{M_d}})^T \sim \text{Dir}(\cdot|\boldsymbol{\alpha}^{(d)})$
8 For each i in $\{1, \ldots, N_d\}$:
9 Generate $z_i \in \{\lambda_1^{(d)}, \ldots, \lambda_{M_d}^{(d)}\} \sim \text{Mult}(\cdot|\boldsymbol{\theta}^{(d)})$
10 Generate $w_i \in \{1, \ldots, V\} \sim \text{Mult}(\cdot|\boldsymbol{\beta}_{z_i})$

Table 1: Generative process for Labeled LDA: $\boldsymbol{\beta}_k$ is a vector consisting of the parameters of the multinomial distribution corresponding to the k^{th} topic, $\boldsymbol{\alpha}$ are the parameters of the Dirichlet topic prior and $\boldsymbol{\eta}$ are the parameters of the word prior, while Φ_k is the label prior for topic k. For the meaning of the projection matrix $L^{(d)}$, please refer to Eq 1.

its labels $\boldsymbol{\Lambda}^{(d)}$. Since the word-topic assignments z_i (see step 9 in Table 1) are drawn from this distribution, this restriction ensures that all the topic assignments are limited to the document's labels.

Towards this objective, we first generate the document's labels $\boldsymbol{\Lambda}^{(d)}$ using a Bernoulli coin toss for each topic k, with a labeling prior probability Φ_k, as shown in step 5. Next, we define the vector of document's labels to be $\boldsymbol{\lambda}^{(d)} = \{k|\Lambda_k^{(d)} = 1\}$. This allows us to define a document-specific label projection matrix $L^{(d)}$ of size $M_d \times K$ for each document d, where $M_d = |\boldsymbol{\lambda}^{(d)}|$, as follows: For each row $i \in \{1, \ldots, M_d\}$ and column $j \in \{1, \ldots, K\}$:

$$L_{ij}^{(d)} = \begin{cases} 1 & \text{if } \lambda_i^{(d)} = j \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

In other words, the i^{th} row of $L^{(d)}$ has an entry of 1 in column j if and only if the i^{th} document label $\lambda_i^{(d)}$ is equal to the topic j, and zero otherwise. As the name indicates, we use the $L^{(d)}$ matrix to project the parameter vector of the Dirichlet topic prior $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)^T$ to a lower dimensional vector $\boldsymbol{\alpha}^{(d)}$ as follows:

$$\boldsymbol{\alpha}^{(d)} = L^{(d)} \times \boldsymbol{\alpha} = (\alpha_{\lambda_1^{(d)}}, \ldots, \alpha_{\lambda_{M_d}^{(d)}})^T \quad (2)$$

Clearly, the dimensions of the projected vector correspond to the topics represented by the labels of the document. For example, suppose $K = 4$ and that a document d has labels given by $\boldsymbol{\Lambda}^{(d)} = \{0, 1, 1, 0\}$ which implies $\boldsymbol{\lambda}^{(d)} = \{2, 3\}$, then $L^{(d)}$ would be:

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Then, $\boldsymbol{\theta}^{(d)}$ is drawn from a Dirichlet distribution with parameters $\boldsymbol{\alpha}^{(d)} = L^{(d)} \times \boldsymbol{\alpha} = (\alpha_2, \alpha_3)^T$ (i.e., with the Dirichlet restricted to the topics 2 and 3).

This fulfills our requirement that the document's topics are restricted to its own labels. The projection step constitutes the deterministic step 6 in Table 1. The remaining part of the model from steps 7 through 10 are the same as for regular LDA.

The dependency of $\boldsymbol{\theta}$ on both $\boldsymbol{\alpha}$ and $\boldsymbol{\Lambda}$ is indicated by directed edges from $\boldsymbol{\Lambda}$ and $\boldsymbol{\alpha}$ to $\boldsymbol{\theta}$ in the plate notation in Figure 1. This is the only additional dependency we introduce in LDA's representation (please compare with Figure 1 in (Blei et al., 2003)).

2.1 Learning and inference

In most applications discussed in this paper, we will assume that the documents are multiply tagged with human labels, both at learning and inference time.

When the labels $\boldsymbol{\Lambda}^{(d)}$ of the document are observed, the labeling prior Φ is d-separated from the rest of the model given $\boldsymbol{\Lambda}^{(d)}$. Hence the model is same as traditional LDA, except the constraint that the topic prior $\boldsymbol{\alpha}^{(d)}$ is now restricted to the set of labeled topics $\boldsymbol{\lambda}^{(d)}$. Therefore, we can use collapsed Gibbs sampling (Griffiths and Steyvers, 2004) for training where the sampling probability for a topic for position i in a document d in Labeled LDA is given by:

$$P(z_i = j|\mathbf{z}_{-i}) \propto \frac{n_{-i,j}^{w_i} + \eta_{w_i}}{n_{-i,j}^{(\cdot)} + \boldsymbol{\eta}^T\mathbf{1}} \times \frac{n_{-i,j}^{(d)} + \alpha_j}{n_{-i,\cdot}^{(d)} + \boldsymbol{\alpha}^T\mathbf{1}} \quad (3)$$

where $n_{-i,j}^{w_i}$ is the count of word w_i in topic j, that does not include the current assignment z_i, a missing subscript or superscript (e.g. $n_{-i,j}^{(\cdot)}$)) indicates a summation over that dimension, and $\mathbf{1}$ is a vector of 1's of appropriate dimension.

Although the equation above looks exactly the same as that of LDA, we have an important distinction in that, the target topic j is restricted to belong to the set of labels, i.e., $j \in \boldsymbol{\lambda}^{(d)}$.

Once the topic multinomials $\boldsymbol{\beta}$ are learned from the training set, one can perform inference on any new labeled test document using Gibbs sampling

restricted to its tags, to determine its per-word label assignments $\mathbf{z}$. In addition, one can also compute its posterior distribution $\boldsymbol{\theta}$ over topics by appropriately normalizing the topic assignments $\mathbf{z}$.

It should now be apparent to the reader how the new model addresses some of the problems in multi-labeled corpora that we highlighted in Section 1. For example, since there is a one-to-one correspondence between the labels and topics, the model can display automatic topical summaries for each label k in terms of the topic-specific distribution $\boldsymbol{\beta}_k$. Similarly, since the model assigns a label z_i to each word w_i in the document d automatically, we can now extract portions of the document relevant to each label k (it would be all words $w_i \in \mathbf{w}^{(d)}$ such that $z_i = k$). In addition, we can use the topic distribution $\boldsymbol{\theta}^{(d)}$ to rank the user specified labels in the order of their relevance to the document, thereby also eliminating spurious ones if necessary.

Finally, we note that other less restrictive variants of the proposed L-LDA model are possible. For example, one could consider a version that allows topics that do not correspond to the label set of a given document with a small probability, or one that allows a common background topic in all documents. We did implement these variants in our preliminary experiments, but they did not yield better performance than L-LDA in the tasks we considered. Hence we do not report them in this paper.

2.2 Relationship to Naive Bayes

The derivation of the algorithm so far has focused on its relationship to LDA. However, Labeled LDA can also be seen as an extension of the event model of a traditional Multinomial Naive Bayes classifier (McCallum and Nigam, 1998) by the introduction of a mixture model. In this section, we develop the analogy as another way to understand L-LDA from a supervised perspective.

Consider the case where no document in the collection is assigned two or more labels. Now for a particular document d with label l_d, Labeled LDA draws each word's topic variable z_i from a multinomial constrained to the document's label set, i.e. $z_i = l_d$ for each word position i in the document. During learning, the Gibbs sampler will assign each z_i to l_d while incrementing $\beta_{l_d}(w_i)$, effectively counting the occurences of each word type in documents labeled with l_d. Thus in the singly labeled document case, the probability of each document under Labeled LDA is equal to the probability of the document under the Multinomial Naive Bayes event model trained on those same document instances. Unlike the Multinomial Naive Bayes classifier, Labeled LDA does not encode a decision boundary for unlabeled documents by comparing $P(\mathbf{w}^{(d)}|l_d)$ to $P(\mathbf{w}^{(d)}|\neg l_d)$, although we discuss using Labeled LDA for multi-label classification in Section 7.

Labeled LDA's similarity to Naive Bayes ends with the introduction of a second label to any document. In a traditional one-versus-rest Multinomial Naive Bayes model, a separate classifier for each label would be trained on all documents with that label, so each word can contribute a count of 1 to every observed label's word distribution. By contrast, Labeled LDA assumes that each document is a mixture of underlying topics, so the count mass of single word instance must instead be distributed over the document's observed labels.

3 Credit attribution within tagged documents

Social bookmarking websites contain millions of tags describing many of the web's most popular and useful pages. However, not all tags are uniformly appropriate at all places within a document. In the sections that follow, we examine mechanisms by which Labeled LDA's credit assignment mechanism can be utilized to help support browsing and summarizing tagged document collections.

To create a consistent dataset for experimenting with our model, we selected 20 tags of medium to high frequency from a collection of documents dataset crawled from *del.icio.us*, a popular social bookmarking website (Heymann et al., 2008). From that larger dataset, we selected uniformly at random four thousand documents that contained at least one of the 20 tags, and then filtered each document's tag set by removing tags not present in our tag set. After filtering, the resulting corpus averaged 781 non-stop words per document, with each document having 4 distinct tags on average. In contrast to many existing text datasets, our tagged corpus is highly multiply labeled: almost 90% of of the documents have more than one tag. (For comparison, less than one third of the news documents in the popular RCV1-v2 collection of newswire are multiply labeled). We will refer to

this collection of data as the del.icio.us tag dataset.

4 Topic Visualization

A first question we ask of Labeled LDA is how its topics compare with those learned by traditional LDA on the same collection of documents. We ran our implementations of Labeled LDA and LDA on the del.icio.us corpus described above. Both are based on the standard collapsed Gibbs sampler, with the constraints for Labeled LDA implemented as in Section 2.

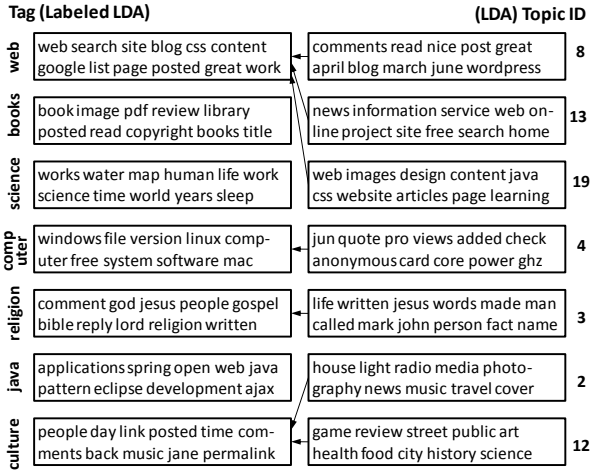


Figure 2: Comparison of some of the 20 topics learned on del.icio.us by Labeled LDA (left) and traditional LDA (right), with representative words for each topic shown in the boxes. Labeled LDA's topics are named by their associated tag. Arrows from right-to-left show the mapping of LDA topics to the closest Labeled LDA topic by cosine similarity. Tags not shown are: *design*, *education*, *english*, *grammar*, *history*, *internet*, *language*, *philosophy*, *politics*, *programming*, *reference*, *style*, *writing*.

Figure 2 shows the top words associated with 20 topics learned by Labeled LDA and 20 topics learned by unsupervised LDA on the del.icio.us document collection. Labeled LDA's topics are directly named with the tag that corresponds to each topic, an improvement over standard practice of inferring the topic name by inspection (Mei et al., 2007). The topics learned by the unsupervised variant were matched to a Labeled LDA topic highest cosine similarity.

The topics selected are representative: compared to Labeled LDA, unmodified LDA allocates many topics for describing the largest parts of the corpus and under-represents tags that are less uncommon: of the 20 topics learned, LDA learned multiple topics mapping to each of five tags (*web*, *culture*, and *computer*, *reference*, and *politics*, all of which were common in the dataset) and learned no topics that aligned with six tags (*books*, *english*, *science*, *history*, *grammar*, *java*, and *philosophy*, which were rarer).

The Elements of Style, William Strunk, Jr.

Asserting that one must first know the rules to break them, this classic reference book is a must-have for any student and conscientious writer. Intended for use in which the practice of composition is combined with the study of literature, it gives in brief space the principal requirements of plain English style and concentrates attention on the rules of usage and principles of composition most commonly violated.

Figure 3: Example document with important words annotated with four of the page's tags as learned by Labeled LDA. Red (single underline) is *style*, green (dashed underline) *grammar*, blue (double underline) *reference*, and black (jagged underline) *education*.

5 Tagged document visualization

In addition to providing automatic summaries of the words best associated with each tag in the corpus, Labeled LDA's credit attribution mechanism can be used to augment the view of a single document with rich contextual information about the document's tags.

Figure 3 shows one web document from the collection, a page describing a guide to writing English prose. The 10 most common tags for that document are *writing, reference, english, grammar, style, language, books, book, strunk*, and *education*, the first eight of which were included in our set of 20 tags. In the figure, each word that has high posterior probability from one tag has been annotated with that tag. The red words come from the *style* tag, green from the *grammar* tag, blue from the *reference* tag, and black from the *education* tag. In this case, the model does very well at assigning individual words to the tags that, subjectively, seem to strongly imply the presence of that tag on this page. A more polished rendering could add subtle visual cues about which parts of a page are most appropriate for a particular set of tags.

books

L-LDA this classic reference book is a must-have for any student and conscientious writer. Intended for

SVM the rules of usage and principles of composition most commonly violated. Search: CONTENTS Bibliographic

language

L-LDA the beginning of a sentence must refer to the grammatical subject 8. Divide words at

SVM combined with the study of literature, it gives in brief space the principal requirements of

grammar

L-LDA requirements of plain English style and concentrates attention on the rules of usage and principles of

SVM them, this classic reference book is a must-have for any student and conscientious writer.

Figure 4: Representative snippets extracted by L-LDA and tag-specific SVMs for the web page shown in Figure 3.

6 Snippet Extraction

Another natural application of Labeled LDA's credit assignment mechanism is as a means of selecting snippets of a document that best describe its contents from the perspective of a particular tag. Consider again the document in Figure 3. Intuitively, if this document were shown to a user interested in the tag *grammar*, the most appropriate snippet of words might prefer to contain the phrase "rules of usage," whereas a user interested in the term *style* might prefer the title "Elements of Style."

To quantitatively evaluate Labeled LDA's performance at this task, we constructed a set of 29 recently tagged documents from del.icio.us that were labeled with two or more tags from the 20 tag subset, resulting in a total of 149 (document,tag) pairs. For each pair, we extracted a 15-word window with the highest tag-specific score from the document. Two systems were used to score each window: Labeled LDA and a collection of one-vs-rest SVMs trained for each tag in the system. L-LDA scored each window as the expected probability that the tag had generated each word. For SVMs, each window was taken as its own document and scored using the tag-specific SVM's un-thresholded scoring function, taking the window with the most positive score. While a complete solution to the tag-specific snippet extraction problem might be more informed by better linguistic features (such as phrase boundaries), this experimental setup suffices to evaluate both kinds of models for their ability to appropriately assign words to underlying labels.

Model	Best Snippet	Unanimous
L-LDA	**72 / 149**	24 / 51
SVM	21 / 149	2 / 51

Table 2: Human judgments of tag-specific snippet quality as extracted by L-LDA and SVM. The center column is the number of document-tag pairs for which a system's snippet was judged superior. The right column is the number of snippets for which all three annotators were in complete agreement (numerator) in the subset of document scored by all three annotators (denominator).

Figure 3 shows some example snippets output by our system for this document. Note that while SVMs did manage to select snippets that were vaguely on topic, Labeled LDA's outputs are generally of superior subjective quality. To quantify this intuition, three human annotators rated each pair of snippets. The outputs were randomly labeled as "System A" or "System B," and the annotators were asked to judge which system generated a better tag-specific document subset. The judges were also allowed to select neither system if there was no clear winner. The results are summarized in Table 2.

L-LDA was judged superior by a wide margin: of the 149 judgments, L-LDA's output was selected as preferable in 72 cases, whereas SVM's was selected in only 21. The difference between these scores was highly significant ($p < .001$) by the sign test. To quantify the reliability of the judgments, 51 of the 149 document-tag pairs were labeled by all three annotators. In this group, the judgments were in substantial agreement,[1] with Fleiss' Kappa at .63.

Further analysis of the triply-annotated subset yields further evidence of L-LDA's advantage over SVM's: 33 of the 51 were tag-page pairs where L-LDA's output was picked by at least one annotator as a better snippet (although L-LDA might not have been picked by the other annotators). And of those, 24 were unanimous in that

[1] Of the 15 judgments that were in contention, only two conflicted on *which* system was superior (L-LDA versus SVM); the remaining disagreements were about whether or not one of the systems was a clear winner.

all three judges selected L-LDA's output. By contrast, only 10 of the 51 were tag-page pairs where SVMs' output was picked by at least one annotator, and of those, only 2 were selected unanimously.

7 Multilabeled Text Classification

In the preceding section we demonstrated how Labeled LDA's credit attribution mechanism enabled effective modeling within documents. In this section, we consider whether L-LDA can be adapted as an effective multi-label classifier for documents as a whole. To answer that question, we applied a modified variant of L-LDA to a multi-label document classification problem: given a training set consisting of documents with multiple labels, predict the set of labels appropriate for each document in a test set.

Multi-label classification is a well researched problem. Many modern approaches incorporate label correlations (e.g., Kazawa et al. (2004), Ji et al. (2008)). Others, like our algorithm are based on mixture models (such as Ueda and Saito (2003)). However, we are aware of no methods that trade off label-specific word distributions with document-specific label distributions in quite the same way.

In Section 2, we discussed learning and inference when labels are observed. In the task of multilabel classification, labels are available at training time, so the learning part remains the same as discussed before. However, inferring the best set of labels for an unlabeled document at test time is more complex: it involves assessing all label assignments and returning the assignment that has the highest posterior probability. However, this is not straight-forward, since there are 2^K possible label assignments. To make matters worse, the support of $\boldsymbol{\alpha}(\Lambda^{(d)})$ is different for different label assignments. Although we are in the process of developing an efficient sampling algorithm for this inference, for the purposes of this paper we make the simplifying assumption that the model reduces to standard LDA at inference, where the document is free to sample from any of the K topics. This is a reasonable assumption because allowing the model to explore the whole topic space for each document is similar to exploring all possible label assignments. The document's most likely labels can then be inferred by suitably thresholding its posterior probability over topics.

As a baseline, we use a set of multiple one-vs-rest SVM classifiers which is a popular and extremely competitive baseline used by most previous papers (see (Kazawa et al., 2004; Ueda and Saito, 2003) for instance). We scored each model based on Micro-F1 and Macro-F1 as our evaluation measures (Lewis et al., 2004). While the former allows larger classes to dominate its results, the latter assigns an equal weight to all classes, providing us complementary information.

7.1 Yahoo

We ran experiments on a corpus from the Yahoo directory, modeling our experimental conditions on the ones described in (Ji et al., 2008).[2] We considered documents drawn from 8 top level categories in the Yahoo directory, where each document can be placed in any number of subcategories. The results were mixed, with SVMs ahead on one measure: Labeled LDA beat SVMs on five out of eight datasets on MacroF1, but didn't win on any datasets on MicroF1. Results are presented in Table 3.

Because only a processed form of the documents was released, the Yahoo dataset does not lend itself well to error analysis. However, only 33% of the documents in each top-level category were applied to more than one sub-category, so the credit assignment machinery of L-LDA was unused for the majority of documents. We therefore ran an artificial second set of experiments considering only those documents that had been given more than one label in the training data. On these documents, the results were again mixed, but Labeled LDA comes out ahead. For MacroF1, L-LDA beat SVMs on four datasets, SVMs beat L-LDA on one dataset, and three were a statistical tie.[3] On MicroF1, L-LDA did much better than on the larger subset, outperforming on four datasets with the other four a statistical tie.

It is worth noting that the Yahoo datasets are skewed by construction to contain many documents with highly overlapping content: because each collection is within the same super-class such as "Arts", "Business", etc., each sub-categories'

[2] We did not carefully tune per-class thresholds of each of the one vs. rest classifiers in each model, but instead tuned only one threshold for all classifiers in each model via cross-validation on the Arts subsets. As such, our numbers were on an average 3-4% less than those reported in (Ji et al., 2008), but the methods were comparably tuned.

[3] The difference between means of multiple runs were not significantly different by two-tailed paired t-test.

Dataset	%MacroF1		%MicroF1	
	L-LDA	SVM	L-LDA	SVM
Arts	30.70(1.62)	23.23 (0.67)	39.81(1.85)	48.42 (0.45)
Business	30.81(0.75)	22.82 (1.60)	67.00(1.29)	72.15 (0.62)
Computers	27.55(1.98)	18.29 (1.53)	48.95(0.76)	61.97 (0.54)
Education	33.78(1.70)	36.03 (1.30)	41.19(1.48)	59.45 (0.56)
Entertainment	39.42(1.38)	43.22 (0.49)	47.71(0.61)	62.89 (0.50)
Health	45.36(2.00)	47.86 (1.72)	58.13(0.43)	72.21 (0.26)
Recreation	37.63(1.00)	33.77 (1.17)	43.71(0.31)	59.15 (0.71)
Society	27.32(1.24)	23.89 (0.74)	42.98(0.28)	52.29 (0.67)

Table 3: Averaged performance across ten runs of multi-label text classification for predicting subsets of the named Yahoo directory categories. Numbers in parentheses are standard deviations across runs. L-LDA outperforms SVMs on 5 subsets with MacroF1, but on no subsets with MicroF1.

vocabularies will naturally overlap a great deal. L-LDA's credit attribution mechanism is most effective at partitioning semantically distinct words into their respective label vocabularies, so we expect that Labeled-LDA's performance as a text classifier would improve on collections with more semantically diverse labels.

7.2 Tagged Web Pages

We also applied our method to text classification on the del.icio.us dataset, where the documents are naturally multiply labeled (more than 89%) and where the tags are less inherently similar than in the Yahoo subcategories. Therefore we expect Labeled LDA to do better credit assignment on this subset and consequently to show improved performance as a classifier, and indeed this is the case.

We evaluated L-LDA and multiple one-vs-rest SVMs on 4000 documents with the 20 tag subset described in Section 3. L-LDA and multiple one-vs-rest SVMs were trained on the first 80% of documents and evaluated on the remaining 20%, with results averaged across 10 random permutations of the dataset. The results are shown in Table 4. We tuned the SVMs' shared cost parameter $C(= 10.0)$ and selected raw term frequency over tf-idf weighting based on 4-fold cross-validation on 3,000 documents drawn from an independent permutation of the data. For L-LDA, we tuned the shared parameters of threshold and proportionality constants in word and topic priors. L-LDA and SVM have very similar performance on MacroF1, while L-LDA substantially outperforms on MicroF1. In both cases, L-LDA's improvement is statistically significantly by a 2-tailed paired t-test at 95% confidence.

Model	%MacroF1	%MicroF1
L-LDA	**39.85** (.989)	**52.12** (.434)
SVM	39.00 (.423)	39.33 (.574)

Table 4: Mean performance across ten runs of multi-label text classification for predicting 20 tags on del.icio.us data. L-LDA outperforms SVMs significantly on both metrics by a 2-tailed, paired t-test at 95% confidence.

8 Discussion

One of the main advantages of L-LDA on multiply labeled documents comes from the model's document-specific topic mixture θ. By explicitly modeling the importance of each label in the document, Labeled LDA can effective perform some contextual word sense disambiguation, which suggests why L-LDA can outperform SVMs on the del.icio.us dataset.

As a concrete example, consider the excerpt of text from the *del.icio.us* dataset in Figure 5. The document itself has several tags, including *design* and *programming*. Initially, many of the likelihood probabilities $p(w|label)$ for the (content) words in this excerpt are higher for the label *programming* than *design*, including "content", "client", "CMS" and even "designed", while *design* has higher likelihoods for just "website" and "happy". However, after performing inference on this document using L-LDA, the inferred document probability for *design* ($p(design)$) is much higher than it is for *programming*. In fact, the higher probability for the tag more than makes up the difference in the likelihood for all the words except "CMS" (Content Management System), so

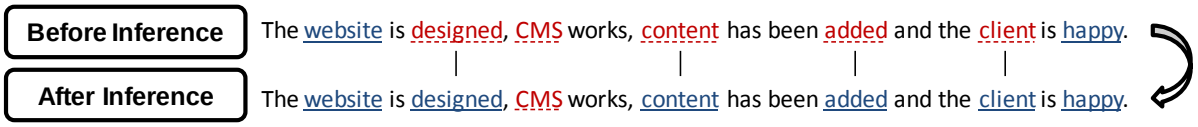


Figure 5: The effect of tag mixture proportions for credit assignment in a web document. Blue (single underline) words are generated from the *design* tag; red (dashed underline) from the *programming* tag. By themselves, most words used here have a higher probability in *programming* than in *design*. But because the document as a whole is more about *design* than *programming*(incorporating words not shown here), inferring the document's topic-mixture θ enables L-LDA to correctly re-assign most words.

that L-LDA correctly infers that most of the words in this passage have more to do with *design* than *programming*.

9 Conclusion

This paper has introduced Labeled LDA, a novel model of multi-labeled corpora that directly addresses the credit assignment problem. The new model improves upon LDA for labeled corpora by gracefully incorporating user supervision in the form of a one-to-one mapping between topics and labels. We demonstrate the model's effectiveness on tasks related to credit attribution within documents, including document visualizations and tag-specific snippet extraction. An approximation to Labeled LDA is also shown to be competitive with a strong baseline (multiple one vs-rest SVMs) for multi-label classification.

Because Labeled LDA is a graphical model in the LDA family, it enables a range of natural extensions for future investigation. For example, the current model does not capture correlations between labels, but such correlations might be introduced by composing Labeled LDA with newer state of the art topic models like the Correlated Topic Model (Blei and Lafferty, 2006) or the Pachinko Allocation Model (Li and McCallum, 2006). And with improved inference for unsupervised $\mathbf{\Lambda}$, Labeled LDA lends itself naturally to modeling semi-supervised corpora where labels are observed for only some documents.

Acknowledgments

This project was supported in part by the President of Stanford University through the IRiSS Initiatives Assessment project.

References

D. M. Blei and J. Lafferty. 2006. Correlated Topic Models. *NIPS*, 18:147.

D. Blei and J McAuliffe. 2007. Supervised Topic Models. In *NIPS*, volume 21.

D. M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent Dirichlet allocation. *JMLR*.

T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *PNAS*, 1:5228–35.

P. Heymann, G. Koutrika, and H. Garcia-Molina. 2008. Can social bookmarking improve web search. In *WSDM*.

S. Ji, L. Tang, S. Yu, and J. Ye. 2008. Extracting shared subspace for multi-label classification. In *KDD*, pages 381–389, New York, NY, USA. ACM.

H. Kazawa, H. Taira T. Izumitani, and E. Maeda. 2004. Maximal margin labeling for multi-topic text categorization. In *NIPS*.

S. Lacoste-Julien, F. Sha, and M. I. Jordan. 2008. DiscLDA: Discriminative learning for dimensionality reduction and classification. In *NIPS*, volume 22.

D. D. Lewis, Y. Yang, T. G. Rose, G. Dietterich, F. Li, and F. Li. 2004. RCV1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397.

Wei Li and Andrew McCallum. 2006. Pachinko allocation: Dag-structured mixture models of topic correlations. In *International conference on Machine learning*, pages 577–584.

A. McCallum and K. Nigam. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 7.

Q. Mei, X. Shen, and C Zhai. 2007. Automatic labeling of multinomial topic models. In *KDD*.

D. Ramage, P. Heymann, C. D. Manning, and H. Garcia-Molina. 2009. Clustering the tagged web. In *WSDM*.

N. Ueda and K. Saito. 2003. Parametric mixture models for multi-labeled text includes models that can be seen to fit within a dimensionality reduction framework. In *NIPS*.

Clustering to Find Exemplar Terms for Keyphrase Extraction

Zhiyuan Liu, Peng Li, Yabin Zheng, Maosong Sun
Department of Computer Science and Technology
State Key Lab on Intelligent Technology and Systems
National Lab for Information Science and Technology
Tsinghua University, Beijing 100084, China
{lzy.thu, pengli09, yabin.zheng}@gmail.com, sms@tsinghua.edu.cn

Abstract

Keyphrases are widely used as a brief summary of documents. Since manual assignment is time-consuming, various unsupervised ranking methods based on importance scores are proposed for keyphrase extraction. In practice, the keyphrases of a document should not only be statistically important in the document, but also have a good coverage of the document. Based on this observation, we propose an unsupervised method for keyphrase extraction. Firstly, the method finds exemplar terms by leveraging clustering techniques, which guarantees the document to be semantically covered by these exemplar terms. Then the keyphrases are extracted from the document using the exemplar terms. Our method outperforms sate-of-the-art graph-based ranking methods (TextRank) by 9.5% in F1-measure.

1 Introduction

With the development of Internet, information on the web is emerging exponentially. How to effectively seek and manage information becomes an important research issue. Keyphrases, as a brief summary of a document, provide a solution to help organize, manage and retrieve documents, and are widely used in digital libraries and information retrieval.

Keyphrases in articles of journals and books are usually assigned by authors. However, most articles on the web usually do not have human-assigned keyphrases. Therefore, automatic keyphrase extraction is an important research task. Existing methods can be divided into supervised and unsupervised approaches.

The supervised approach (Turney, 1999) regards keyphrase extraction as a classification task. In this approach, a model is trained to determine whether a candidate term of the document is a keyphrase, based on statistical and linguistic features. For the supervised keyphrase extraction approach, a document set with human-assigned keyphrases is required as training set. However, human labelling is time-consuming. Therefore, in this study we focus on unsupervised approach.

As an example of an unsupervised keyphrase extraction approach, the graph-based ranking (Mihalcea and Tarau, 2004) regards keyphrase extraction as a ranking task, where a document is represented by a term graph based on term relatedness, and then a graph-based ranking algorithm is used to assign importance scores to each term. Existing methods usually use term cooccurrences within a specified window size in the given document as an approximation of term relatedness (Mihalcea and Tarau, 2004).

As we know, none of these existing works gives an explicit definition on what are appropriate keyphrases for a document. In fact, the existing methods only judge the importance of each term, and extract the most important ones as keyphrases.

From the observation of human-assigned keyphrases, we conclude that good keyphrases of a document should satisfy the following properties:

1. **Understandable**. The keyphrases are understandable to people. This indicates the extracted keyphrases should be grammatical. For example, “machine learning” is a grammatical phrase, but “machine learned” is not.

2. **Relevant**. The keyphrases are semantically relevant with the document theme. For example, for a document about “machine learning”, we want the keyphrases all about this theme.

3. **Good coverage**. The keyphrases should

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 257–266,
Singapore, 6-7 August 2009.

cover the whole document well. Suppose we have a document describing "Beijing" from various aspects of "location", "atmosphere" and "culture", the extracted keyphrases should cover all the three aspects, instead of just a partial subset of them.

The classification-based approach determines whether a term is a keyphrase in isolation, which could not guarantee Property 3. Neither does the graph-based approach guarantee the top-ranked keyphrases could cover the whole document. This may cause the resulting keyphrases to be inappropriate or badly-grouped.

To extract the appropriate keyphrases for a document, we suggest an unsupervised clustering-based method. Firstly the terms in a document are grouped into clusters based on semantic relatedness. Each cluster is represented by an exemplar term, which is also the centroid of each cluster. Then the keyphrases are extracted from the document using these exemplar terms.

In this method, we group terms based on semantic relatedness, which guarantees a good coverage of the document and meets Property 2 and 3. Moreover, we only extract the keyphrases in accordance with noun group (chunk) patterns, which guarantees the keyphrases satisfy Property 1.

Experiments show that the clustering-based method outperforms the state-of-the-art graph-based approach on precision, recall and F1-measure. Moreover, this method is unsupervised and language-independent, which is applicable in the web era with enormous information.

The rest of the paper is organized as follows. In Section 2, we introduce and discuss the related work in this area. In Section 3, we give an overview of our method for keyphrase extraction. From Section 4 to Section 7, the algorithm is described in detail. Empirical experiment results are demonstrated in Section 8, followed by our conclusions and plans for future work in Section 9.

2 Related Work

A straightforward method for keyphrase extraction is to select keyphrases according to frequency criteria. However, the poor performance of this method drives people to explore other methods. A pioneering achievement is carried out in (Turney, 1999), as mentioned in Section 1, a supervised machine learning method was suggested in this paper which regards keyphrase extraction as a classification task. In this work, parameterized heuristic rules are combined with a genetic algorithm into a system for keyphrase extraction. A different learning algorithm, Naive Bayes method, is applied in (Frank et al., 1999) with improved results on the same data used in (Turney, 1999). Hulth (Hulth, 2003; Hulth, 2004) adds more linguistic knowledge, such as syntactic features, to enrich term representation, which significantly improves the performance. Generally, the supervised methods need manually annotated training set, which may sometimes not be practical, especially in the web scenario.

Starting with TextRank (Mihalcea and Tarau, 2004), graph-based ranking methods are becoming the most widely used unsupervised approach for keyphrase extraction. The work in (Litvak and Last, 2008) applies HITS algorithm on the word graph of a document under the assumption that the top-ranked nodes should be the document keywords. Experiments show that classification-based supervised method provides the highest keyword identification accuracy, while the HITS algorithm gets the highest F-measure. Work in (Huang et al., 2006) also considers each document as a term graph where the structural dynamics of these graphs can be used to identify keyphrases. Wan and Xiao (Wan and Xiao, 2008b) use a small number of nearest neighbor documents to provide more knowledge to improve graph-based keyphrase extraction algorithm for single document. Motivated by similar idea, Wan and Xiao (Wan and Xiao, 2008a) propose to adopt clustering methods to find a small number of similar documents to provide more knowledge for building word graphs for keyword extraction. Moreover, after our submission of this paper, we find that a method using community detection on semantic term graphs is proposed for keyphrase extraction from multi-theme documents (Grineva et al., 2009). In addition, some practical systems, such as KP-Miner (Elbeltagy and Rafea, 2009), also do not need to be trained on a particular human-annotated document set.

In recent years, a number of systems are developed for extracting keyphrases from web documents (Kelleher and Luz, 2005; Chen et al., 2005), email (Dredze et al., 2008) and some other specific sources, which indicates the importance of keyphrase extraction in the web era. However,

none of these previous works has overall consideration on the essential properties of appropriate keyphrases mentioned in Section 1.

We should also note that, although the precision and recall of most current keyphrase extractors are still much lower compared to other NLP-tasks, it does not indicate the performance is poor because even different annotators may assign different keyphrases to the same document. As described in (Wan and Xiao, 2008b), when two annotators were asked to label keyphrases on 308 documents, the Kappa statistic for measuring inter-agreement among them was only 0.70.

3 Algorithm Overview

The method proposed in this paper is mainly inspired by the nature of appropriate keyphrases mentioned in Section 1, namely *understandable*, semantically *relevant* with the document and *high coverage* of the whole document.

Let's analyze the document describing "Beijing" from the aspects of "location", "atmosphere" and "culture". Under the bag-of-words assumption, each term in the document, except for function words, is used to describe an aspect of the theme. Based on these aspects, terms are grouped into different clusters. The terms in the same cluster are more relevant with each other than with the ones in other clusters. Taking the terms "temperature", "cold" and "winter" for example, they may serve the aspect "atmosphere" instead of "location" or some other aspects when talking about "Beijing".

Based on above description, it is thus reasonable to propose a clustering-based method for keyphrase extraction. The overview of the method is:

1. **Candidate term selection.** We first filter out the stop words and select candidate terms for keyphrase extraction.

2. **Calculating term relatedness.** We use some measures to calculate the semantic relatedness of candidate terms.

3. **Term clustering.** Based on term relatedness, we group candidate terms into clusters and find the exemplar terms of each cluster.

4. **From exemplar terms to keyphrases.** Finally, we use these exemplar terms to extract keyphrases from the document.

In the next four sections we describe the algorithm in detail.

4 Candidate Term Selection

Not all words in a document are possible to be selected as keyphrases. In order to filter out the noisy words in advance, we select candidate terms using some heuristic rules. This step proceeds as follows. Firstly the text is tokenized for English or segmented into words for Chinese and other languages without word-separators. Then we remove the stop words and consider the remaining single terms as candidates for calculating semantic relatedness and clustering.

In methods like (Turney, 1999; Elbeltagy and Rafea, 2009), candidate keyphrases were first found using n-gram. Instead, in this method, we just find the single-word terms as the candidate terms at the beginning. After identifying the exemplar terms within the candidate terms, we extract multi-word keyphrases using the exemplars.

5 Calculating Term Relatedness

After selecting candidate terms, it is important to measure term relatedness for clustering. In this paper, we propose two approaches to calculate term relatedness: one is based on term cooccurrence within the document, and the other by leveraging human knowledge bases.

5.1 Cooccurrence-based Term Relatedness

An intuitive method for measuring term relatedness is based on term cooccurrence relations within the given document. The cooccurrence relation expresses the cohesion relationships between terms.

In this paper, cooccurrence-based relatedness is simply set to the count of cooccurrences within a window of maximum w words in the whole document. In the following experiments, the window size w is set from 2 to 10 words.

Each document can be regarded as a word sequence for computing cooccurrence-based relatedness. There are two types of word sequence for counting term cooccurrences. One is the original word sequence without filtering out any words, and the other is after filtering out the stop words or the words with specified part-of-speech (POS) tags. In this paper we select the first type because each word in the sequence takes important role for measuring term cooccurrences, no matter whether

it is a stop word or something else. If we filter out some words, the term relatedness will not be as precise as before.

In experiments, we will investigate how the window size influences the performance of keyphrase extraction.

5.2 Wikipedia-based Term Relatedness

Many methods have been proposed for measuring the relatedness between terms using external resources. One principled method is leveraging human knowledge bases. Inspired by (Gabrilovich and Markovitch, 2007), we adopt Wikipedia, the largest encyclopedia collected and organized by human on the web, as the knowledge base to measure term relatedness.

The basic idea of computing term relatedness by leveragting Wikipedia is to consider each Wikipedia article as a concept. Then the semantic meaning of a term could be represented as a weighted vector of Wikipedia concepts, of which the values are the term's TFIDF within corresponding Wikipedia articles. We could compute the term relatedness by comparing the concept vectors of the terms. Empirical evaluations confirm that the idea is effective and practical for computing term relatedness (Gabrilovich and Markovitch, 2007).

In this paper, we select cosine similarity, Euclidean distance, Point-wise Mutual Information and Normalized Google Similarity Distance (Cilibrasi and Vitanyi, 2007) for measuring term relatedness based on the vector of Wikipedia concepts.

Denote the Wikipedia-concept vector of the term t_i as $C_i = \{c_{i1}, c_{i2}, ..., c_{iN}\}$, where N indicates the number of Wikipedia articles, and c_{ik} is the TFIDF value of w_i in the kth Wikipedia article. The cosine similarity is defined as

$$cos(i,j) = \frac{C_i \cdot C_j}{\|C_i\|\|C_j\|} \tag{1}$$

The definition of Euclidean distance is

$$euc(i,j) = \sqrt{\sum_{k=1}^{N}(c_{ik} - c_{jk})^2} \tag{2}$$

Point-wise Mutual Information (PMI) is a common approach to quantify relatedness. Here we take three ways to measure term relatedness using PMI. One is based on Wikipedia page count,

$$pmi_p(i,j) = \log_2 \frac{N \times p(i,j)}{p(i) \times p(j)} \tag{3}$$

where $p(i,j)$ is the number of Wikipedia articles containing both t_i and t_j, while $p(i)$ is the number of articles which contain t_i. The second is based on the term count in Wikipedia articles,

$$pmi_t(i,j) = \log_2 \frac{T \times t(i,j)}{t(i) \times t(j)} \tag{4}$$

where T is the number of terms in Wikipedia, $t(i,j)$ is the number of t_i and t_j occurred adjacently in Wikipedia, and $t(i)$ is the number of t_i in Wikipedia. The third one is a combination of the above two PMI ways,

$$pmi_c(i,j) = \log_2 \frac{N \times pt(i,j)}{p(i) \times p(j)} \tag{5}$$

where $pt(i,j)$ indicates the number of Wikipedia articles containing t_i and t_j as adjacency. It is obvious that $pmi_c(i,j) \leq pmi_p(i,j)$, and $pmi_c(i,j)$ is more strict and accurate for measuring relatedness.

Normalized Google Similarity Distance (NGD) is a new measure for measuring similarity between terms proposed by (Cilibrasi and Vitanyi, 2007) based on information distance and Kolmogorov complexity. It could be applied to compute term similarity from the World Wide Web or any large enough corpus using the page counts of terms. NGD used in this paper is based on Wikipedia article count, defined as

$$ngd(i,j) = \frac{\max(\log p(i), \log p(j)) - logp(i,j)}{\log N - \min(logp(i), logp(j))} \tag{6}$$

where N is the number of Wikipedia articles used as normalized factor.

Once we get the term relatedness, we could then group the terms using clustering techniques and find exemplar terms for each cluster.

6 Term Clustering

Clustering is an important unsupervised learning problem, which is the assignment of objects into groups so that objects from the same cluster are more similar to each other than objects from different clusters (Han and Kamber, 2005). In this paper, we use three widely used clustering algorithms, hierarchical clustering, spectral clustering and Affinity Propagation, to cluster the candidate terms of a given document based on the semantic relatedness between them.

6.1 Hierarchical Clustering

Hierarchical clustering groups data over a variety of scales by creating a cluster tree. The tree is a multilevel hierarchy, where clusters at one level are joined as clusters at the next level. The hierarchical clustering follows this procedure:

1. Find the distance or similarity between every pair of data points in the dataset;
2. Group the data points into a binary and hierarchical cluster tree;
3. Determine where to cut the hierarchical tree into clusters. In hierarchical clustering, we have to specify the cluster number m in advance.

In this paper, we use the hierarchical clustering implemented in Matlab Statistics Toolbox. Note that although we use hierarchical clustering here, the cluster hierarchy is not necessary for the clustering-based method.

6.2 Spectral Clustering

In recent years, spectral clustering has become one of the most popular modern clustering algorithms. Spectral clustering makes use of the spectrum of the similarity matrix of the data to perform dimensionality reduction for clustering into fewer dimensions, which is simple to implement and often outperforms traditional clustering methods such as k-means. Detailed introduction to spectral clustering could be found in (von Luxburg, 2006).

In this paper, we use the spectral clustering toolbox developed by Wen-Yen Chen, et al. (Chen et al., 2008) [1]. Since the cooccurrence-based term relatedness is usually sparse, the traditional eigenvalue decomposition in spectral clustering will sometimes get run-time error. In this paper, we use the singular value decomposition (SVD) technique for spectral clustering instead.

For spectral clustering, two parameters are required to be set by the user: the cluster number m, and σ which is used in computing similarities from object distances

$$s(i,j) = \exp(\frac{-d(i,j)^2}{2\sigma^2}) \quad (7)$$

where $s(i,j)$ and $d(i,j)$ are the similarity and distance between i and j respectively.

[1]The package could be accessed via `http://www.cs.ucsb.edu/~wychen/sc.html`.

6.3 Affinity Propagation

Another powerful clustering method, Affinity Propagation, is based on message passing techniques. AP was proposed in (Frey and Dueck, 2007), where AP was reported to find clusters with much lower error than those found by other methods. In this paper, we use the toolbox developed by Frey, et al. [2].

Detailed description of the algorithm could be found in (Frey and Dueck, 2007). Here we introduced three parameters for AP:

- **Preference**. Rather than requiring predefined number of clusters, Affinity Propagation takes as input a real number p for each term, so that the terms with larger p are more likely to be chosen as exemplars, i.e., centroids of clusters. These values are referred to as "preferences". The preferences are usually be set as the maximum, minimum, mean or median of $s(i,j), i \neq j$.
- **Convergence criterion**. AP terminates if (1) the local decisions stay constant for I_1 iterations; or (2) the number of iterations reaches I_2. In this work, we set I_1 to 100 and I_2 to $1,000$.
- **Damping factor**. When updating the messages, it is important to avoid numerical oscillations by using damping factor. Each message is set to λ times its value from the previous iteration plus $1-\lambda$ times its prescribed updated value, where the damping factor λ is between 0 and 1. In this paper we set $\lambda = 0.9$.

7 From Exemplar Terms to Keyphrases

After term clustering, we select the exemplar terms of each clusters as seed terms. In Affinity Propagation, the exemplar terms are directly obtained from the clustering results. In hierarchical clustering, exemplar terms could also be obtained by the Matlab toolbox. While in spectral clustering, we select the terms that are most close to the centroid of a cluster as exemplar terms.

As reported in (Hulth, 2003), most manually assigned keyphrases turn out to be noun groups. Therefore, we annotate the document with POS

[2]The package could be accessed via `http://www.psi.toronto.edu/affinitypropagation/`.

tags using Stanford Log-Linear Tagger [3], and then extract the noun groups whose pattern is zero or more adjectives followed by one or more nouns. The pattern can be represented using regular expressions as follows

$$(JJ) * (NN|NNS|NNP)+$$

where JJ indicates adjectives and various forms of nouns are represented using NN, NNS and NNP. From these noun groups, we select the ones that contain one or more exemplar terms to be the keyphrases of the document.

In this process, we may find single-word keyphrases. In practice, only a small fraction of keyphrases are single-word. Thus, as a part of postprocessing process, we have to use a frequent word list to filter out the terms that are too common to be keyphrases.

8 Experiment Results

8.1 Datasets and Evaluation Metric

The dataset used in the experiments is a collection of scientific publication abstracts from the *Inspec* database and the corresponding manually assigned keyphrases [4]. The dataset is used in both (Hulth, 2003) and (Mihalcea and Tarau, 2004). Each abstract has two kinds of keyphrases: controlled keyphrases, restricted to a given dictionary, and uncontrolled keyphrases, freely assigned by the experts. We use the uncontrolled keyphrases for evaluation as proposed in (Hulth, 2003) and followed by (Mihalcea and Tarau, 2004).

As indicated in (Hulth, 2003; Mihalcea and Tarau, 2004), in uncontrolled manually assigned keyphrases, only the ones that occur in the corresponding abstracts are considered in evaluation. The extracted keyphrases of various methods and manually assigned keyphrases are compared after stemming.

In the experiments of (Hulth, 2003), for her supervised method, Hulth splits a total of $2,000$ abstracts into $1,000$ for training, 500 for validation and 500 for test. In (Mihalcea and Tarau, 2004), due to the unsupervised method, only the test set was used for comparing the performance of TextRank and Hulth's method.

For computing Wikipedia-based relatedness, we use a snapshot on November 11, 2005 [5]. The frequent word list used in the postprocessing step for filtering single-word phrases is also computed from Wikipedia. In the experiments of this paper, we add the words that occur more than $1,000$ times in Wikipedia into the list.

The clustering-based method is completely unsupervised. Here, we mainly run our method on test set and investigate the influence of relatedness measurements and clustering methods with different parameters. Then we compare our method with two baseline methods: Hulth's method and TextRank. Finally, we analyze and discuss the performance of the method by taking the abstract of this paper as a demonstration.

8.2 Influence of Relatedness Measurements

We first investigate the influence of semantic relatedness measurements. By systematic experiments, we find that Wikipedia-based relatedness outperforms cooccurrence-based relatedness for keyphrase extraction, though the improvement is not significant. In Table 1, we list the performance of spectral clustering with various relatedness measurements for demonstration. In this table, the w indicates the window size for counting cooccurrences in cooccurrence-based relatedness. cos, euc, etc. are different measures for computing Wikipedia-based relatedness which we presented in Section 5.2.

Table 1: Influence of relatedness measurements for keyphrase extraction.

Parameters	Precision	Recall	F1-measure
Cooccurrence-based Relatedness			
$w = 2$	0.331	0.626	0.433
$w = 4$	0.333	0.621	0.434
$w = 6$	0.331	0.630	0.434
$w = 8$	0.330	0.623	0.432
$w = 10$	0.333	0.632	0.436
Wikipedia-based Relatedness			
cos	0.348	0.655	0.455
euc	0.344	0.634	0.446
pmi_p	0.344	0.621	0.443
pmi_t	0.344	0.619	0.442
pmi_c	0.350	0.660	**0.457**
ngd	0.343	0.620	0.442

[3] The package could be accessed via `http://http://nlp.stanford.edu/software/tagger.shtml`.

[4] Many thanks to Anette Hulth for providing us the dataset.

[5] The dataset could be get from `http://www.cs.technion.ac.il/~gabr/resources/code/wikiprep/`.

We use spectral clustering here because it outperforms other clustering techniques, which will be shown in the next subsection. The results in Table 1 are obtained when the cluster number $m = \frac{2}{3}n$, where n is the number of candidate terms obtained in Section 5. Besides, for Euclidean distance and Google distance, we set $\sigma = 36$ of Formula 7 to convert them to corresponding similarities, where we get the best result when we conduct different trails with $\sigma = 9, 18, 36, 54$, though there are only a small margin among them.

As shown in Table 1, although the method using Wikipedia-based relatedness outperforms that using cooccurrence-based relatedness, the improvement is not prominent. Wikipedia-based relatedness is computed according to global statistical information on Wikipedia. Therefore it is more precise than cooccurrence-based relatedness, which is reflected in the performance of the keyphrase extraction. However, on the other hand, Wikipedia-based relatedness does not catch the document-specific relatedness, which is represented by the cooccurrence-based relatedness. It will be an interesting future work to combine these two types of relatedness measurements.

From this subsection, we conclude that, although the method using Wikipedia-based relatedness performs better than cooccurrence-based one, due to the expensive computation of Wikipedia-based relatedness, the cooccurrence-based one is good enough for practical applications.

8.3 Influence of Clustering Methods and Their Parameters

To demonstrate the influence of clustering methods for keyphrase extraction, we fix the relatedness measurement as Wikipedia-based pmi_c, which has been shown in Section 8.2 to be the best relatedness measurement.

In Table 2, we show the performance of three clustering techniques for keyphrase extraction. For hierarchical clustering and spectral clustering, the cluster number m are set explicitly as the proportion of candidate terms n, while for Affinity Propagation, we set preferences as the minimum, mean, median and maximum of $s(i, j)$ to get different number of clusters, denoted as min, $mean$, $median$ and max in the table respectively.

As shown in the table, when cluster number m is large, spectral clustering outperforms hierarchical clustering and Affinity Propagation. Among these methods, only Affinity Propagation under some parameters performs poorly.

Table 2: Influence of clustering methods for keyphrase extraction.

Parameters	Precision	Recall	F1-measure
Hierarchical Clustering			
$m = \frac{1}{4}n$	0.365	0.369	0.367
$m = \frac{1}{3}n$	0.365	0.369	0.367
$m = \frac{1}{2}n$	0.351	0.562	0.432
$m = \frac{2}{3}n$	0.346	0.629	0.446
$m = \frac{4}{5}n$	0.340	0.657	0.448
Spectral Clustering			
$m = \frac{1}{4}n$	0.385	0.409	0.397
$m = \frac{1}{3}n$	0.374	0.497	0.427
$m = \frac{1}{2}n$	0.374	0.497	0.427
$m = \frac{2}{3}n$	0.350	0.660	**0.457**
$m = \frac{4}{5}n$	0.340	0.679	0.453
Affinity Propagation			
$p = max$	0.331	0.688	0.447
$p = mean$	0.433	0.070	0.121
$p = median$	0.422	0.078	0.132
$p = min$	0.419	0.059	0.103

8.4 Comparing with Other Algorithms

Table 3 lists the results of the clustering-based method compared with the best results reported in (Hulth, 2003; Mihalcea and Tarau, 2004) on the same dataset. For each method, the table lists the total number of assigned keyphrases, the mean number of keyphrases per abstract, the total number of correct keyphrases, and the mean number of correct keyphrases. The table also lists precision, recall and F1-measure. In this table, hierarchical clustering, spectral clustering and Affinity Propagation are abbreviated by "HC", "SC" and "AP" respectively.

The result of Hulth's method listed in this table is the best one reported in (Hulth, 2003) on the same dataset. This is a supervised classification-based method, which takes more linguistic features in consideration for keyphrase extraction. The best result is obtained using n-gram as candidate keyphrases and adding POS tags as candidate features for classification.

The result of TextRank listed here is the best one reported in (Mihalcea and Tarau, 2004) on the same dataset. To obtain the best result, the authors built an undirected graph using window $w = 2$ on word sequence of the given document, and ran

Table 3: Comparison results of Hulth's method, TextRank and our clustering-based method.

	Assigned		Correct				
Method	Total	Mean	Total	Mean	Precision	Recall	F1-measure
Hulth's	7,815	15.6	1,973	3.9	0.252	0.517	0.339
TextRank	6,784	13.7	2,116	4.2	0.312	0.431	0.362
HC	7,303	14.6	2,494	5.0	0.342	0.657	0.449
SC	7,158	14.3	2,505	5.0	**0.350**	0.660	**0.457**
AP	8,013	16.0	2,648	5.3	0.330	**0.697**	0.448

PageRank on it.

In this table, the best result of hierarchical clustering is obtained by setting the cluster number $m = \frac{2}{3}n$ and using Euclidean distance for computing Wikipedia-based relatedness. The parameters of spectral clustering are the same as in last subsection. For Affinity Propagation, the best result is obtained under $p = max$ and using Wikipedia-based Euclidean distance as relatedness measure.

From this table, we can see clustering-based method outperforms TextRank and Hulth's method. For spectral clustering, F1-measure achieves an approximately 9.5% improvement as compared to TextRank.

Furthermore, since the clustering-based method is unsupervised, we do not need any set for training and validation. In this paper, we also carry out an experiment on the whole Hulth's dataset with $2,000$ abstracts. The performance is similar to that on 500 abstracts as shown above. The best result is obtained when we use spectral clustering by setting $m = \frac{2}{3}n$ with Wikipedia-based pmi_c relatedness, which is the same in 500 abstracts. In this result, we extract $29,517$ keyphrases, among which $9,655$ are correctly extracted. The precision, recall and F1-measure are 0.327, 0.653 and 0.436 respectively. The experiment results show that the clustering-based method is stable.

8.5 Analysis and Discussions

From the above experiment results, we can see the clustering-based method is both robust and effective for keyphrase extraction as an unsupervised method.

Here, as an demonstration, we use spectral clustering and Wikipedia-based pmi_c relatedness to extract keyphrases from the abstract of this paper. The extracted stemmed keyphrases under various cluster numbers are shown in Figure 1. In this figure, we find that when $m = \frac{1}{4}n, \frac{1}{3}n, \frac{1}{2}n$, the extracted keyphrases are identical, where the exemplar terms under $m = \frac{1}{3}n$ are marked in boldface. We find several aspects like "unsupervised", "exemplar term" and "keyphrase extraction" are extracted correctly. In fact, "clustering technique" in the abstract should also be extracted as a keyphrase. However, since "clustering" is tagged as a verb that ends in -ing, which disagrees the noun group patterns, thus the phrase is not among the extracted keyphrases.

When $m = \frac{2}{3}n$, the extracted keyphrases are noisy with many single-word phrases. As the cluster number increases, more exemplar terms are identified from these clusters, and more keyphrases will be extracted from the document based on exemplar terms. If we set the cluster number to $m = n$, all terms will be selected as exemplar terms. In this extreme case, all noun groups will be extracted as keyphrases, which is obviously not proper for keyphrase extraction. Thus, it is important for this method to appropriately specify the cluster number.

In the experiments, we also notice that frequent word list is important for keyphrase extraction. Without the list for filtering, the best F1-measure will decrease by about 5 percent to 40%. However, the solution of using frequent word list is somewhat too simple, and in future work, we plan to investigate a better combination of clustering-based method with traditional methods using term frequency as the criteria.

9 Conclusion and Future Work

In this paper, we propose an unsupervised clustering-based keyphrase extraction algorithm. This method groups candidate terms into clusters and identify the exemplar terms. Then keyphrases are extracted from the document based on the exemplar terms. The clustering based on term semantic relatedness guarantees the extracted keyphrases have a good coverage of the document. Experiment results show the method has a good ef-

Figure 1: Keyphrases in stemmed form extracted from this paper's abstract.

Keyphrases when $m = \frac{1}{4}n, \frac{1}{3}n, \frac{1}{2}n$
unsupervis method; various **unsupervis** rank method; **exemplar term**; state-of-the-art **graph-bas** rank method; **keyphras**; **keyphras** extract
Keyphrases when $m = \frac{2}{3}n$
unsupervis method; manual assign; brief summari; various unsupervis rank method; exemplar term; document; state-of-the-art graph-bas rank method; experi; keyphras; import score; keyphras extract

fectiveness and robustness, and outperforms baselines significantly.

Future work may include:

1. Investigate the feasibility of clustering directly on noun groups;

2. Investigate the feasibility of combining cooccurrence-based and Wikipedia-based relatedness for clustering;

3. Investigate the performance of the method on other types of documents, such as long articles, product reviews and news;

4. The solution of using frequent word list for filtering out too common single-word keyphrases is undoubtedly simple, and we plan to make a better combination of the clustering-based method with traditional frequency-based methods for keyphrase extraction.

Acknowledgments

This work is supported by the National 863 Project under Grant No. 2007AA01Z148 and the National Science Foundation of China under Grant No. 60621062. The authors would like to thank Anette Hulth for kindly sharing her datasets.

References

Mo Chen, Jian-Tao Sun, Hua-Jun Zeng, and Kwok-Yan Lam. 2005. A practical system of keyphrase extraction for web pages. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 277–278.

Wen Y. Chen, Yangqiu Song, Hongjie Bai, Chih J. Lin, and Edward Chang. 2008. Psc: Paralel spectral clustering. Submitted.

Rudi L. Cilibrasi and Paul M. B. Vitanyi. 2007. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383.

Mark Dredze, Hanna M. Wallach, Danny Puller, and Fernando Pereira. 2008. Generating summary keywords for emails using topics. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 199–206.

S. Elbeltagy and A. Rafea. 2009. Kp-miner: A keyphrase extraction system for english and arabic documents. *Information Systems*, 34(1):132–144.

Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 668–673.

Brendan J J. Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science*.

E. Gabrilovich and S. Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 6–12.

M. Grineva, M. Grinev, and D. Lizorkin. 2009. Extracting key terms from noisy and multi-theme documents. In *Proceedings of the 18th international conference on World wide web*, pages 661–670. ACM New York, NY, USA.

Jiawei Han and Micheline Kamber. 2005. *Data Mining: Concepts and Techniques, second edition*. Morgan Kaufmann.

Chong Huang, Yonghong Tian, Zhi Zhou, Charles X. Ling, and Tiejun Huang. 2006. Keyphrase extraction using semantic networks structure analysis. In *Proceedings of the 6th International Conference on Data Mining*, pages 275–284.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223.

A. Hulth. 2004. Reducing false positives by expert combination in automatic keyword indexing. *Recent Advances in Natural Language Processing III: Selected Papers from RANLP 2003*, page 367.

Daniel Kelleher and Saturnino Luz. 2005. Automatic hypertext keyphrase detection. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*.

Marina Litvak and Mark Last. 2008. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop Multi-source Multilingual Information Extraction and Summarization*, pages 17–24.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Peter D. Turney. 1999. Learning to Extract Keyphrases from Text. *National Research Council Canada, Institute for Information Technology, Technical Report ERB-1057*.

U. von Luxburg. 2006. A tutorial on spectral clustering. Technical report, Max Planck Institute for Biological Cybernetics.

Xiaojun Wan and Jianguo Xiao. 2008a. Collabrank: Towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of COLING*, pages 969–976.

Xiaojun Wan and Jianguo Xiao. 2008b. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 855–860.

Geo-mining: Discovery of Road and Transport Networks Using Directional Patterns

Dmitry Davidov
ICNC
The Hebrew University of Jerusalem
dmitry@alice.nc.huji.ac.il

Ari Rappoport
Institute of Computer Science
The Hebrew University of Jerusalem
arir@cs.huji.ac.il

Abstract

One of the most desired information types when planning a trip to some place is the knowledge of transport, roads and geographical connectedness of prominent sites in this place. While some transport companies or repositories make some of this information accessible, it is not easy to find, and the majority of information about uncommon places can only be found in web free text such as blogs and forums. In this paper we present an algorithmic framework which allows an automated acquisition of map-like information from the web, based on surface patterns like "from X to Y". Given a set of locations as initial seeds, we retrieve from the web an extended set of locations and produce a map-like network which connects these locations using transport type edges. We evaluate our framework in several settings, producing meaningful and precise connection sets.

1 Introduction

Textual geographical information such as location descriptions, directions, travel guides and transport tables is extensively used by people. Discovering such information automatically can assist NLP applications such as question answering (Santos and Cardoso, 2008), and can be useful for a variety of other applications, including automatic map annotation.

Some textual geographical information can be found in web sites of transport companies, tourist sites and repositories such as Wikipedia. Such sites usually utilize structured information such as machine-readable meta-data, tables, schedule forms or lists, which are relatively convenient for processing. However, automatic utilization of such information is limited. Even on these sites, only a small fraction of the available geographical information is stored in a well-structured and freely accessible form. With the growth of the web, information can be frequently found in 'ordinary' web pages such as forums, travelogues or news. In such sites, information is usually noisy, unstructured and present in the form of free text.

This type of information can be addressed by lexical patterns. Patterns were shown to be very useful in all sorts of lexical acquisition tasks, giving high precision results at relatively low computational costs (Pantel et al., 2004). Pattern-driven search engine queries allow to access such information and gather the required data very efficiently (Davidov et al., 2007).

In this paper we present a framework that given a few seed locations as a specification of a region, discovers additional locations (including alternate location names) and map-like travel paths through this region labeled by transport type labels.

The type of output produced by our framework here differs from that in previous pattern-based studies. Unlike mainstream pattern-based web mining, it does not target some specific two-slot relationship and attempts to extract word tuples for this relationship. Instead, it discovers geographical networks of transport or access connections. Such networks are not unstructured sets of word pairs, but a structured graph with labeled edges.

Our algorithm utilizes variations of the basic pre-defined pattern "[Transport] from Location1 to Location2" which allows location names and connections to be captured starting from the given seed location set. We acquire search engine snippets and extract contexts where location names co-appear. Next we construct a location graph and merge transport edges to identify main transport group types. Finally, we improve the obtained data by reducing transitive connections and identifying key locations.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 267–275,
Singapore, 6-7 August 2009.

The obtained location data can be used as a draft for preparation of travel resources and on-demand travel plans. It can also be used for question answering systems and for automated enrichment and verification of existing geographical resources.

We evaluate our framework on three different regions of different scale and type: Annapurna in Nepal, the south Israel area and the Cardiff area in England. In our evaluation we estimated precision and the amount of discovered locations and transport edges, and examined the quality of the obtained map as a whole by visually comparing the overall connectedness of the graph to an actual road or transport map.

2 Related Work

In this paper we utilize a pattern-based lexical acquisition framework for the discovery of geographical information. Due to the importance of lexical databases for many NLP tasks, substantial research has been done on direct or indirect automated acquisition of concepts (sets of terms sharing a significant aspect of their meaning) and concept relationships in the form of graphs connecting concepts or terms inside concepts into usually hierarchical or bipartite networks. In the case of geo-mining, concepts can include sets of alternative names for some place, or sets of all locations of the same type (e.g., all countries). Geographical relationships can include nearness of two locations and entity-location relationships such as institute-address, capital-country, tourist site-city etc.

The major differences between relationship acquisition frameworks come from the types and annotation requirements of the supplied input and the basic algorithmic approach used to process this input. A first major algorithmic approach is to represent word contexts as vectors in some space and use distributional measures and automatic clustering in that space. Curran (2002) and Lin (1998) use syntactic features in the vector definition. Caraballo (1999) uses conjunction and appositive annotations in the vector representation.While efforts have been made for improving the computational complexity of these methods (Gorman and Curran, 2006), they remain data and computation intensive.

The second major algorithmic approach is to use lexico-syntactic patterns, which have been shown to produce more accurate results than feature vectors at a lower computational cost on large corpora (Pantel et al., 2004). Most related work deals with discovery of hypernymy (Hearst, 1992; Pantel and Lin, 2002) and synonymy (Widdows and Dorow, 2002; Davidov and Rappoport, 2006). Some studies deal with the discovery of more specific relation sub-types, including inter-verb relations (Chklovski and Pantel, 2004) and semantic relations between nominals (Davidov and Rappoport, 2008). Extensive frameworks were proposed for iterative discovery of pre-specified (e.g., (Riloff and Jones, 1999)) and unspecified (e.g., (Agichtein and Gravano, 2000)) relation types.

Some concepts and relationships examined by seed-based discovery methods were of a geographical nature. For example, (Etzioni et al., 2004) discovered a set of countries and (Davidov et al., 2007) discovered diverse country relationships, including location relationships between a country and its capital and a country and its rivers. As noted in Section 1, the type of output that we produce here is not an unstructured collection of word pairs but a labeled network. As such, our task here is much more complex.

Our study is related to geographical information retrieval (GIR) systems. However, our problem is very far from classic GIR problem settings. In GIR, the goal is to classify or retrieve possibly multilingual documents in response to queries in the form 'theme, spatial relationship, location', e.g., 'mountains near New York' (Purves et al., 2006). Our goal, in contrast, is not document retrieval, but the generation of a structured information resource, a labeled location graph.

Spatial relationships used in natural language tend to be qualitative and descriptive rather than quantitative. The concept of Naive Geography, which reflects the way people think and write about geographical space, is described in (Egenhofer and Shariff, 1995). Later in (Egenhofer and Shariff, 1998) they proposed a way to convert coordinate-based relationships between spatial entities to natural language using terms as 'crosses', 'goes through' or 'runs into'. Such terms can be potentially used in patterns to extract geographical information from text. In this paper we start from a different pattern, 'from ... to', which helps in discovering transport or connectedness relationships between places, e.g., 'bus from X to Y' and 'road from X to Y'.

The majority of geographical data mining

frameworks utilize structured data such as available gazetteers and Wikipedia metadata. Several other studies utilize semi-structured data like Wikipedia links (Overell and Ruger, 2007) or substructures in web pages, including addresses and phone numbers (Borges et al., 2007).

The recent Schockaert et al.(2008) framework for extraction of topological relationships from the web has some similarities to our study. In both cases the algorithm produces map-like structures using the web. However, there are significant differences. They utilize relatively structured address data on web pages and rely on the order of named entities in address data for extracting containment relationships. They also use co-appearances in addresses (e.g., 'R1 / R2' and 'R1 & R2' as in "Newport & Gabalfa, Cardiff") to deduce location boundaries. This allows them to get high precision data for modern and heavily populated regions like Cardiff, where the majority of offices have available well-formatted web pages.

However, in less populated regions (a major target for tourist information requests), this strategy could be problematic since a major information source about these places would be not local web sites (in which local addresses are likely to be found) but foreign visitor sites, web forums and news. We rely on free text available in all types of web pages, which allows us to capture unstructured information which contains a significant portion of the web-available geographical knowledge.

Our goals are also different from Schockaert et al.(2008), since we focus on obtaining information based on paths and transport between locations, while in their work the goal is to find a network representing nearness of places rather than their connectivity by means of transport or walking. Nevertheless, in one of our evaluation settings we targeted the area of Cardiff as in (Schockaert et al., 2008). This allowed us to make an indirect comparison of a relevant part of our results to previous work, achieving state-of-the-art performance.

3 The Algorithm

As input to our algorithm we are given a seed of a few location names specifiying some geographical region. In this section we describe the algorithm which, given these names, extracts the labeled structure of connections between entities in the desired region. We first use a predefined pattern for recursive extraction of the first set of entities. Then we discover additional patterns from co-appearing location pairs and use them to get more terms. Next, we label and merge the obtained location pairs. Finally, we construct and refine the obtained graph.

3.1 Pattern-based discovery with web queries

In order to obtain the first set of location connections, we use derivatives of the basic pattern 'from X to Y'. Using Yahoo! BOSS, we have utilized the API's ability to search for phrases with wildcards. Given a location name L we start the search with patterns "from * to L", "from * * to L". These are Yahoo! BOSS queries where enclosing words in "" means searching for an exact phrase and '*' means a wildcard for exactly one arbitrary word.

This pattern serves a few goals beyond the discovery of connectedness. Thus putting '*'s inside the pattern rather than using "from L to" allowed us to avoid arbitrary truncation of multiword expressions as in 'from Moscow to St. Petersburg' and reduced the probability of capturing unrelated sentence parts like 'from Moscow to cover deficit'.

Location names are usually ambiguous and this type of web queries can lead to a significant amount of noise or location mix. There are two types of ambiguity. First, as in 'from Billericay to Stock....', stock can be a location or a verb. We filter most such cases by allowing only capitalized location names. Besides, such an ambiguity is rarely captured by 'from * to L' patterns. The second type is location ambiguity. Thus 'Moscow' refers to at least 5 location names including farms in Africa and Australia and a locality in Ireland. In order to reduce mixing of locations we use the following simple disambiguation technique. Before performing "from...to" queries, we downloaded up to 100 web pages pointed by each possible *pair* from the given seed locations, generating from a location pair L_1, L_2 a conjunctive query "L_1 * L_2". Then we extracted the most informative terms using a simple probabilistic metric:

$$Rank(w) = \frac{P(w|QueryCorpus)}{P(w|GeneralCorpus)},$$

comparing word distribution in the downloaded $QueryCorpus$ to a large general purpose offline $GeneralCorpus$[1]. We thus obtained a set of

[1]We used the DMOZ corpus (Gabrilovich and Markovitch, 2005).

query-specific disambiguating words. Then we added to all queries the same most frequent word (*DW*) out of the ten words with highest ranks. Thus for the seed set {Moscow, St. Petersburg}, an example of a query is <"from * to Moscow" Russia>.

3.2 Iterative location retrieval

We retrieved all search engine snippets for each of these initial queries[2]. If we succeeded to get more than 50 snippets, we did not download the complete documents. In case where only a handful of snippets were obtained, the algorithm downloaded up to 25 documents pointed by these snippets in an attempt to get more pattern instances. In the majority of tested cases, snippets provide enough information for our task, and this information was not significantly extended by downloading the whole documents.

Once we retrieve snippets we identify terms appearing in the snippets in wildcard slots. For example, if the query is <"from * to Moscow" Russia> and we encounter a snippet "...from Vladivostok to Moscow...", we add 'Vladivostok' to our set of seeds. We then continue the search in a breadth first search setting, stopping the search on three conditions: (1) runaway detected – the total frequency of newly obtained terms through some term's patterns is greater than the total frequency of previously discovered terms+seeds. In this case we stop exploration through the problematic term and continue exploration through other terms[3]; (2) we reached a predefined maximal depth D[4]; (3) no new terms discovered.

At the end of this stage we get the extended set of terms using the set of snippets where these terms co-appear in patterns.

3.3 Enhancement of initial pattern set

In order to get more data, we enhance the pattern set both by discovery of new useful secondary patterns and by narrowing existing patterns. After obtaining the new pattern set we repeat the extraction stage described in Section 3.2.

[2]Yahoo! Boss allows downloading up to a 1000 descriptions, up to 50 in each request. Thus for each seed word, we have performed a few dozen search requests.

[3]Note that the 'problematic' term may be the central term in the region we focus upon – if this happen it means that the seeds do not specify the region well.

[4]Depth is a function of the richness of transport links in the domain. For connected domains (Cardiff, Israel) we used 4, for less connected ones (Nepal) we used 10.

Adding secondary patterns. As in a number of previous studies, we improve our results discovering additional patterns from the obtained term set. The algorithm selects a subset of up to 50 discovered (t_1, t_2) term pairs appearing in 'from t_1 to t_2' patterns and performs the set of additional queries of the form <"t_1 * t_2" DW>.

We then extract from the obtained snippets the patterns of the form 'Prefix t_1 Infix t_2 Postfix', where Prefix and Postfix should contain either a punctuation symbol or 1-3 words. Prefix/Postfix should also be bounded from left/right by a punctuation or one of the 50 most frequent words in the language (based on word counts in the offline general corpus). Infix should contain 1-3 words with the possible addition of punctuation symbols[5].

We examine patterns and select useful ones according to the following ranking scheme, based on how well each pattern captures named entities. For each discovered pattern we scan the obtained snippets and offline general corpus for instances where this pattern connects one of the original or discovered location terms to some other term. Let T be the set of all one to three word terms in the language, $T_d \subset T$ the set of discovered terms, $T_c \subset T$ the set of all capitalized terms and $Pat(t_1, t_2)$ indicates one or more co-appearances of t_1 and t_2 in pattern Pat in the retrieved snippets or offline general corpus. The rank R of pattern Pat is defined by:

$$R(Pat) = \frac{|\{Pat|Pat(t_1, t_2), t_1 \in T_c, t_2 \in T_d\}|}{|\{Pat|Pat(t_1, t_2), t_1 \in T, t_2 \in T_d\}|}$$

In other words, we rank patterns according to the percentage of capitalized words connected by this pattern. We sort patterns by rank and select the top 20% patterns. Once we have discovered a new pattern set, we repeat the term extraction in Section 3.2. We do this only once and not reiterate this loop in order to avoid potential runaway problems. Obtained secondary patterns include different from/to templates "to X from Y by bus"; time/distance combinations "X -N km bus- Y", "X (bus, N min) Y" or patterns in different languages with English location/transport names.

Narrowing existing patterns. When available data volume is high, we would like to take advantage of more data by utilizing more specific pattern

[5]Search engines do not support punctuation in queries, hence these symbols were omitted in web requests and considered only when processing the retrieved snippets.

sets. Since Yahoo! allows to obtain only the first 1K snippets, in case we get more than 10K hits, we extend our queries by adding the most common term sequences appearing before or after the pattern. Thus if for the query "from * to Moscow" we got more than 10K hits and among the snippets we see '... bus from X to Moscow...' we create an extended pattern 'bus from * to Moscow' and use the term extraction in Section 3.2 to get additional terms. Unlike the extraction of secondary patterns, this narrowing process can be repeated recursively as long as a query brings more than 10K results.

3.4 Extraction of labeled connections

At the end of the discovery stage we get an extended set of patterns and a list of search engine snippets discovered using these patterns. Each snippet which captures terms t_1, t_2 in either primary 'from t_1 to t_2' or secondary patterns represents a potential connection between entities.

Using an observed property of the primary pattern, we select as a label a term or set of terms appearing directly before 'from' and delimited with some high frequency word or punctuation. For example, labels for snippets based on 'from...to' patterns and containing 'the road from...', 'got a bus from', 'a TransSiberian train from...' would be road, bus and TransSiberian train.

Once we acquire labels for the primary patterns, we also attempt to find labels in snippets obtained for secondary patterns discovered as described in Section 3.3. We first locate some already labeled pairs in secondary patterns' snippets where we can see both label and the labeled term pair. Then, based on the label's position in this snippet, we define a label slot position for this type of snippet. Suppose that during the labeling of primary pattern snippets we assigned the label 'bus' to the pair (Novgorod, Moscow) and during the pattern extension stage the algorithm discovered a pattern P_{new} = 'ride to t_2 from t_1,' with a corresponding snippet '... getting bus ride to Moscow from Novgorod...'. Then using the labeled pair our algorithm defines the label slot in such a snippet type: 'getting [label] ride to t_2 from t_1'. Once a label slot is defined, all other pairs captured by P_{new} can be successfully labeled.

3.5 Merging connection labels

Some labels may denote the same type of connection. Also, large sets of connections can share the same set of transport types. In this case it is desired to assign a single label for a shared set of transports. We do this by a simple merging technique. Let C_1, C_2 be sets of pairs assigned to labels L_1, L_2. We merge two labels if one of the following conditions holds:

$$(1)|C_1 \cap C_2| > 0.75 * min(|C_1|, |C_2|)$$

$$(2)|C_1 \cap C_2| > 0.45 * max(|C_1|, |C_2|)$$

Thus, either one group is nearly included in the other or each group shares nearly half with the other group. We apply this rule only once and do not iterate recursively. At this stage we also dismiss weakly populated labels, keeping the 10 most populated labels.

3.6 Processing of connection graph

Now once we have merged and assigned the labels we create a pattern graph for each label and attempt to clean the graph of noise and unnecessary edges. Our graph definition follows (Widdows and Dorow, 2002). In our pattern graph for label L, nodes represent terms and directed edges represent co-appearance of two terms in some pattern in snippet labeled by L. We do not add unlabeled snippets to the graph. Now we use a set of techniques to reduce noise and unnecessary edges.

3.6.1 Transitivity elimination

One of the main problems with the pattern-based graph is transitivity of connections. Thus if location A is connected to B and B to C, we frequently acquire a "shortcut" edge connecting A to C. Such an edge diminishes our ability to create a clear and meaningful spatial graph. In order to reduce such edges we employ the following two strategies.

First, neighboring places frequently form fully connected subgraphs. We would like to simplify such cliques to reduce the amount of transitive connections. If three overlapping sets of nodes $\{A_1 \dots A_{n-2}\}$,$\{A_2 \dots A_{n-1}\}$,$\{A_3 \dots A_n\}$ form three different cliques, then we remove all edges between A_1 and the nodes in the third clique and between A_n and the nodes in the first clique.

Second, in paths obtained by directional patterns, it is common that if there is a path $A_1 \rightarrow A_2 \cdots \rightarrow A_n$ where A_1 and A_n are some major 'key' locations[6], then each of the nodes $A_2 \dots A_{n-1}$ tend to be connected both to A_1 and

[6]Such locations will be shown in double circles in the evaluation.

to A_n while intermediate nodes are usually connected only to their close neighbors. We would like to eliminate such transitive edges leaving only the inter-neighbor connections.

We define as key nodes in a graph, nodes whose degree is more than 1.5 times the average graph degree. Then we eliminate the transitive connections: if A_1 is a key node and A_1 is connected to each of the nodes $A_2 \dots A_{n-1}$, and $\forall i \in \{2 \dots n-1\}, A_i$ is connected to A_{i+1}, then we remove the connection of A_1 to all nodes $A_3 \dots A_{n-1}$, leaving A_1 only connected to A_2.

3.6.2 Clearing noise and merging names

Finally we remove potential noise which accidentally connects remote graph parts. If some edge discovered through a single pattern instance connects distant (distance>3) parts of the graph we remove it.

Additionally, we would like to merge common name alternations and misspellings of places. We merge two nodes A and B into one node if either (1) A, B have exactly the same edges, and their edge count is greater than 2; or (2) edges of A are subset of B's edges and the string edit distance between A and B is less than a third of min(StringLength(A), StringLength(B)).

4 Evaluation

Since our problem definition differs significantly from available related work, it is not possible to make direct comparisons. We selected three different cases (in Nepal, Israel, Wales) where the obtained information can be reliably verified, and applied our framework on these settings. As a development set, we used the Russian rail network.

We have estimated the quality of our framework using several measures and observations. First, we calculated the precision and quantity of obtained locations using map information. Then we manually estimated precision of the proposed edges and their labels, comparing them with factual information obtained from maps[7], transport companies and tourist sites. Finally we visually compared a natural drawing of the obtained graph with a real map. In addition, while our goals differ, the third evaluation setting has deliberate significant similarities to (Schockaert et al., 2008), which allows us to make some comparisons.

[7]We recognize that in case of some labels, e.g. 'walk', the precision measure is subjective. Nevertheless it provides a good indication for the quality of our results.

4.1 The Annapurna trek area

One of the most famous sites in Nepal is the Annapurna trekking circuit. This is a 14-21 day walking path which passes many villages. Most of the tourists going through this path spend weeks in prior information mining and preparations. However, even when using the most recent maps and guides, they discover that available geographical knowledge is far from being complete and precise. This trek is a good example of a case when formal information is lacking while free-text shared experience in the web is abundant. Our goal was to test whether the algorithm can discover such knowledge automatically starting from few seed location names (we used Pokhara, which is one of the central cities in the area, and Khudi, a small village). The quality of results for this task was very good. While even crude recall estimation is very hard for this type of task, we have discovered 100% of the Annapurna trek settlements with population over 1K, all of the flight and bus connections, and about 80% of the walking connections.

On Figure 1 we can compare the real map and the obtained map[8]. This discovered map includes a partial map[9] for 4 labels – flights, trek, bus and jeep. You can see on the map different lines for each label. The algorithm discovered 132 entities, all of them Annapurna-related locations. This includes correctly recognized typos and alternative spellings, and the average was 1.2 names per place. For example for Besisahar and Pokhara the following spellings were recognized based both on string distance and spatial collocation: *Besishahar, Bensisahar, BesiSahar, Besi Sahar, Beshishahar, Beisahar, Phokra, Pohkala, Pokahara, Pokhara, Pokhar, Pokra, Pokhura, Pokhra.*

We estimated correctness of edges comparing to existing detailed maps. 95% of the edges were correctly placed and labeled. Results were good since this site is well covered and also not very interconnected – most of it is connected in a single possible way. After the elimination process described in the previous section, only 6% of the nodes participate in 3-cliques. Thus, due to the linearity of the original path, our method success-

[8]Graph nodes were manually positioned such that edges do not intersect. Recall that our goal is to build a network graph, which is an abstract structure. The 2-D embedding of the graph shown here is only for illustrative purposes and is not part of our algorithm.

[9]A few dozens of correctly discovered places were omitted to make the picture readable.

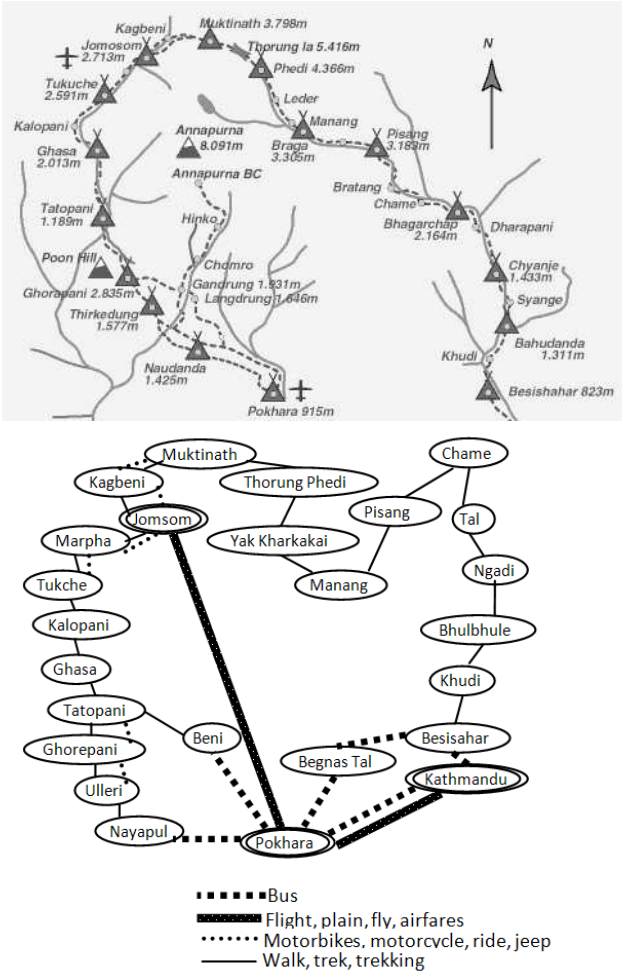


Figure 1: Real path map of Annapurna circuit (above) compared to automatically acquired graph (below). The graph nodes were manually positioned such that edges do not cross each other. Dozens of correctly discovered places were omitted for readability. Double circles indicate key nodes as explained in section 3.6.1

fully avoided the problem of mixing transitively connected nodes into one large clique.

4.2 The Israeli south

The southern part of Israel (mostly the Negev desert) is a sparsely populated region containing a few main roads and a few dozen towns. There is a limited number of tourists sites in the Negev and hence little web information is supposed to be available. Our goal was to see if the algorithm can successfully detect at least major entities and to discover their connectedness.

We discovered 56 names of different places, of them 50 correctly belong to the region, where the region is defined as south from the Ashquelon-Jerusalem-Yericho line, the other 6 were Israeli cities/locations outside the region (Tiberias, Metulla, Ben Gurion, Tel Aviv, Ashdod, Haifa). In addition we discovered 23 alternative names for some of the 56 places. We also constructed the corresponding connectedness graphs.

We tested the usefulness of this data attempting to find the discovered terms in the NGA GEOnet Names Server[10] which is considered one of the most exhaustive geographical resources. We could find in the database only 60% of the correctly discovered English terms denoting towns, so 40% of the terms were discovered by us and ignored by this huge coverage database. We also tested the quality of edges, and found that 80% of the discovered edges were correctly placed and labeled. Figure 2 shows a partial graph of the places obtained for the 'road' label.

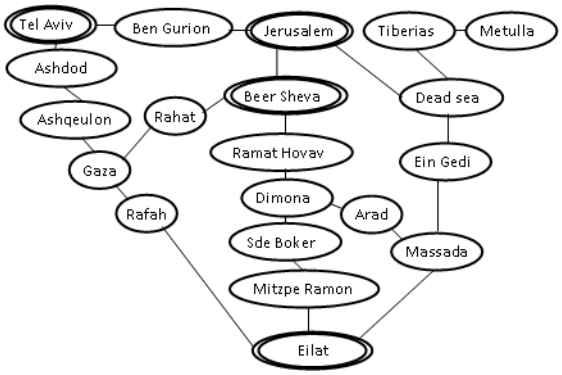


Figure 2: Partial graph for Israel south settings.

4.3 The Cardiff area

Cardiff is the capital, largest city and most populous county in Wales. Our goal was to see if we can discover basic means of transport and corresponding locations connected to and inside Cardiff. This exploration also allowed us to compare some of our results to related studies. We executed our algorithm using as seeds Grangetown, Cardiff and Barry. Table 1 shows the most utilized merged labels obtained for most edge-populated graphs together with graph size and estimated precision. In case of flights, treks and trains, precision was estimated using exact data. In other cases we estimated precision based on reading relevant web pages. We can see that the majority of connectivity sets are meaningful and the precision obtained for most of these sets is high. Figure 3 shows a partial graph for 'walking'-type labels and Figure

[10] http://earth-info.nga.mil/gns/html/

Nodes	Edges(Prec)	Label
88	120(81)	walking,walk,cycling,short ride taxis, Short bus ride,short walk
131	140(95)	flights, airlines,# flights a day
12	16(100)	foot path, trek, walking # miles
36	51(89)	train, railway, rail travel,rail
32	98(65)	bus, road, drive,direct bus

Table 1: The merged labels obtained for 5 most edge-populated graphs, including number of nodes and edges for each label. The estimated precision according to each label definition is shown in parentheses.

4 shows such a graph for train labels[11]. Comparing the obtained map with real map data we notice a definite correlation between actual and induced relative connection of discovered places.

(Schockaert et al., 2008) used their framework to discover neighborhoods of Cardiff. In our case, the most appropriate relation which connects neighborhood locations is walking/cycling. Hence, comparing the results to previous work, we have examined the results obtained for the 'walking' label in details. (Schockaert et al., 2008) report discovery of 68 locations, of them 7 are alternate entries, 4 can be considered vernacular or colloquial, 10 are not considered to be neighborhoods, and 5 are either close to, but not within, Cardiff, or are areas within Cardiff that are not recognized neighborhoods. In our set we have discovered 88 neighborhood names, of them 18 are alternate entries of correct neighborhoods, 4 can be considered vernacular or colloquial, 3 are not considered to be neighborhoods, and 15 are areas outside the Cardiff area.

Considering alternate entries as hits, we got superior precision of $66/88 = 0.75$ in comparison to $49/68 = 0.72$. It should be noted however that we found many more alternative names possibly due to our larger coverage. Also both our framework and the goal were substantially different.

5 Discussion

In this paper we presented a framework which, given a small set of seed terms describing a geographical region, discovers an underlying connectivity and transport graph together with the extraction of common and alternative location names in this region. Our framework is based on the observation that 'from...to'-like patterns can encode connectedness in very precise manner. In our framework, we have combined iterative pattern- and web-based relationship acquisition with the discovery of new patterns and refinement of the location graph. In our evaluation we showed that our framework is capable of extracting high quality non-trivial information from free text given very restricted input and not relying on any heavy preprocessing techniques such as parsing or NER.

The success of the proposed framework opens many challenging directions for its enhancement. Thus we would like to incorporate in our network patterns which allow traveling times and distances to be extracted, such as 'N miles from X to Y'. While in this paper we focus on specific type of geographical relationships, similar frameworks can be useful for a wider class of spatial relationships. Automated acquisition of spatial data can significantly help many NLP tasks, e.g., question answering. We would also like to incorporate some patterns based on (Egenhofer and Shariff, 1998), such as 'crosses', 'goes through' or 'runs into', which may allow automated acquisition of complex spatial relationships. Finally, we would like to incorporate in our framework mod-

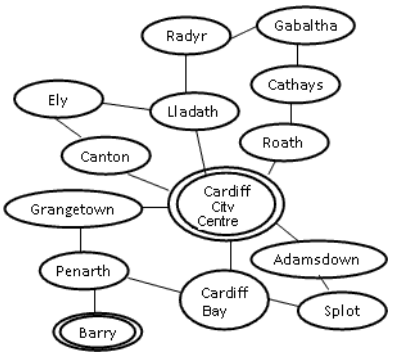


Figure 3: Partial graph of the obtained Cardiff region for the walk/walking/cycling label.

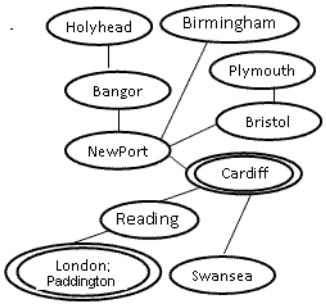


Figure 4: Partial graph of the obtained Cardiff region for the railway/train label.

[11] Spatial position of displayed graph components is arbitrary, we only made sure that there are no intersecting edges.

ules which may allow recognition of structured data, like those developed by (Schockaert et al., 2008).

References

Eugene Agichtein, Luis Gravano, 2000. Snowball: Extracting Relations from Large Plain-text Collections. *ACM DL '00*.

Karla Borges, Alberto Laender, Claudia Medeiros, Clodoveu Davis, 2007. Discovering Geographic Locations in Web Pages Using Urban Addresses. *Fourth Workshop on Geographic Information Retrieval*.

Sharon Caraballo, 1999. Automatic Construction of a Hypernym-labeled Noun Hierarchy from Text. *ACL '99*.

Timothy Chklovski, Patrick Pantel 2004. VerbOcean: Mining the Web for Fine-grained Semantic Verb Relations. *EMNLP '04*.

James R. Curran, Marc Moens, 2002. Improvements in Automatic Thesaurus Extraction. *SIGLEX '02*.

Dmitry Davidov, Ari Rappoport, 2006. Efficient Unsupervised Discovery of Word Categories Using Symmetric Patterns and High Frequency Words. *COLING-ACL '06.*

Dmitry Davidov, Ari Rappoport, Moshe Koppel, 2007. Fully Unsupervised Discovery of Concept-specific Relationships by Web Mining. *ACL '07*.

Dmitry Davidov, Ari Rappoport, 2008. Classification of Semantic Relationships Between Nominals Using Pattern Clusters. *ACL '08.*

Max Egenhofer, Rashid Shariff, 1995. Naive Geography. *Proceedings of COSIT '95.*

Max Egenhofer, Rashid Shariff, 1998. Metric Details for Natural-Language Spatial Relations. *Journal of the ACM TOIS*, 4:295–321, 1998.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, Alexander Yates, 2004. Web-scale Information Extraction in KnowItAll. *WWW '04*.

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

James Gorman, James R. Curran, 2006. Scaling Distributional Similarity to Large Corpora. *COLING-ACL '06*.

Marti Hearst, 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. *COLING '92*.

Dekang Lin, 1998. Automatic Retrieval and Clustering of Similar Words. *COLING '98*.

Simon Overell, Stefan Ruger, 2007. Geographic Co-occurrence as a Tool for GIR. *Fourth ACM Workshop on Geographical information retrieval*.

Patrick Pantel, Dekang Lin, 2002. Discovering Word Senses from Text. *SIGKDD '02*.

Patrick Pantel, Deepak Ravichandran, Eduard Hovy, 2004. Towards Terascale Knowledge Acquisition. *COLING '04*.

Ross Purves, Paul Clough, Christopher Jones, Avi Arampatzis, Benedicte Bucher, Gaihua Fu, Hideo Joho, Awase Syed, Subodh Vaid, Bisheng Yang, 2007. The Design and Implementation of SPIRIT: a Spatially Aware Search Engine for Information Retrieval on the Internet. *International Journal of Geographical Information Science*, 21(7):717-745.

Ellen Riloff, Rosie Jones, 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. *AAAI '99*.

Diana Santos, Nuno Cardoso, 2008. GikiP: Evaluating Geographical Answers from Wikipedia. *Fifth Workshop on Geographic Information Retrieval*.

Steven Schockaert, Philip Smart, Alia Abdelmoty, Christopher Jone, 2008. Mining Topological Relations from the Web. *International Workshop on Flexible Database and Information System Technology*, workshop at DEXA, Turin, pp. 652–656.

Dominic Widdows, Beate Dorow, 2002. A Graph Model for Unsupervised Lexical Acquisition. *COLING '02*.

Wikipedia as Frame Information Repository

Sara Tonelli
FBK-irst
I-38100, Trento, Italy
satonelli@fbk.eu

Claudio Giuliano
FBK-irst
I-38100, Trento, Italy
giuliano@fbk.eu

Abstract

In this paper, we address the issue of automatic extending lexical resources by exploiting existing knowledge repositories. In particular, we deal with the new task of linking FrameNet and Wikipedia using a word sense disambiguation system that, for a given pair frame – lexical unit (F, l), finds the Wikipage that best expresses the the meaning of l. The mapping can be exploited to straightforwardly acquire new example sentences and new lexical units, both for English and for all languages available in Wikipedia. In this way, it is possible to easily acquire good-quality data as a starting point for the creation of FrameNet in new languages. The evaluation reported both for the monolingual and the multilingual expansion of FrameNet shows that the approach is promising.

1 Introduction

Many applications in the context of natural language processing or information retrieval have proved to convey significant improvement by exploiting lexical databases with high-quality annotation such as *FrameNet* (Fillmore et al., 2003) and *WordNet* (Fellbaum, 1998). Nevertheless, the practical use of similar resources is often biased by their limited coverage because manual annotation is time-consuming and requires a relevant financial effort. For this reason, some research activities have focused on the automatic enrichment of such resources with annotated information in (near) manual quality. The main strategy proposed was the mapping between resources in order to reciprocally enrich different lexical databases by linking their information layers. This has proved to be useful in several tasks, from verb classification (Chow and Webster, 2007) to semantic role labeling (Giuglea and Moschitti, 2006), open text semantic parsing (Shi and Mihalcea, 2004) and textual entailment (Burchardt and Frank, 2006).

In this work, we focus on the automatic enrichment of the FrameNet database for English and we propose a new framework to extend this procedure to new languages. While similar works in the past have mainly proposed to automatically extend the FrameNet database by mapping frames and WordNet synsets (Shi and Mihalcea (2005), Johansson and Nugues (2007), and Tonelli and Pighin (2009)), we present an explorative approach that for the first time exploits Wikipedia to this purpose. In particular, given a lexical unit l belonging to a frame F, we devise a strategy to link l to the Wikipedia article that best captures the sense of l in F. This is basically a word disambiguation (WSD) problem (Erk, 2004) and to this purpose we employ a state-of-the-art WSD system (Gliozzo et al., 2005). The mapping between (F, l) pairs and Wikipedia pages could then be exploited for three further subtasks: (a) automatically extract from Wikipedia all sentences pointing to the Wikipage mapped with (F, l) and assign them to F; (b) automatically expand the lexical units sets in the English FrameNet by exploiting the redirecting and linking strategy of Wikipedia; and (c) since Wikipedia is available in 260 languages, use the English Wikipedia article linked to (F, l) as a bridge to carry out sentence and lexical unit retrieval in other languages. The set of automatically collected data would represent the starting point for the creation of FrameNet in new languages. In fact, having a repository of sentences extracted from Wikipedia which have already been divided by sense would significantly speed up the annotation process. In this way, the annotators would not need to extract all sentences in a corpus containing l and classify them by sense. Instead, they should simply validate the given sentences and assign the correct frame elements.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 276–285,
Singapore, 6-7 August 2009.

In the following, we start by providing a brief overview of FrameNet and Wikipedia and we present their structure and organization. Next, we describe the algorithm for mapping lexical units and Wikipages and the word sense disambiguation algorithm employed by the system. In Section 5 we describe the dataset used in the first experiment and report evaluation results of the mapping between (F, l) pairs and Wikipedia senses. In Section 6 we describe an application of the mapping, i.e. the automatic enrichment of English FrameNet. We describe the data extraction process and evaluate the quality of the data. In Section 7 we describe and evaluate another application of the mapping, i.e. the acquisition of data for the automatic creation of Italian FrameNet using the Italian Wikipedia. Finally, we draw conclusions and present future research directions.

2 FrameNet and Wikipedia

FrameNet (Fillmore et al., 2003) is a lexical resource for English based on corpus evidence, whose conceptual model comprises a set of prototypical situations called *frames*, the frame-evoking words or expressions called *lexical units* (LUs) and the roles or participants involved in these situations, called *frame elements*. All lexical units belonging to the same frame have similar semantics but, differently from WordNet synsets, they can belong to different categories and present different parts of speech. For example, the KILLING frame is described in the FrameNet database[1] as *"A Killer or Cause causes the death of the Victim"*. The elements in capitals are the semantic roles (*frame elements*) typically involved in the KILLING situation. The frame definition comes also with the list of frame-evoking lexical units, namely *annihilate.v, annihilation.n, butchery.n, carnage.n, crucify.v, deadly.a*, etc. Since FrameNet is a corpus-based resource, every lexical unit should be instantiated by a set of example sentences, where the frame elements are annotated as well. Instead, FrameNet is still an ongoing project and in the latest release (v. 1.3) there are about 3,380 lexical units out of 10,195 that come with no example sentences. In this work we focus on these lexical units and propose how to automatically collect the missing sentences. Anyhow, the algorithm we propose is suitable also for expanding sentence sets already present in FrameNet.

Wikipedia[2] is one of the largest online repositories of encyclopedic knowledge, with millions of articles available for a large number of languages (>2,800,000 for English). The *article* (or *page*) is the basic entry in Wikipedia. Every article has an unique reference, i.e., one or more words that identify the page and are present in its URL. For example, *Ball_(dance)* identifies the page that describes several types of ball intended as formal dance, while *Dance_(musical_form)* describes the dance as musical genre. Every Wikipedia article is linked to others, and in the body of every page there are plenty of links to connect the most relevant terms to other pages. Another important attribute is the presence of about 3,000,000 redirection pages, that given an identifier that is not present in Wikipedia, automatically display the page with the most semantically similar identifier (for example *Killing* is redirected to the *Murder* page). Wikipedia contains also more than 100,000 disambiguation pages listing all senses (pages) for an ambiguous entity. For example, *Book* has 9 senses, which correspond to 9 different articles. Wikipedia structure and quality make this resource particularly suitable for information extraction and word sense disambiguation tasks (Csomai and Mihalcea (2008) and Milne and Witten (2008)). In fact, page references can be seen as senses and Wikipedia as a large sense inventory. From this point of view, also linking a lexical unit to the correct Wikipedia page is a word sense disambiguation issue because it implies recognizing what meaning the lexical unit has in the given frame. For example, *dance.n* in the SOCIAL_EVENT frame should be linked to *Ball_(dance)* and not to *Dance_(musical_form)*.

3 The Mapping Algorithm

In this section, we describe how to map a frame – lexical unit pair (F, l) into the Wikipedia article that best captures the sense of l as defined in F. The mapping problem is casted as a supervised WSD problem, in which l must be disambiguated using F to provide the context and Wikipedia to provide the sense inventory and the training data. Even if the idea of using Wikipedia links for disambiguation is not novel (Cucerzan, 2007), it is applied for the first time to FrameNet lexical units, considering a frame as a sense definition. The proposed algorithm is summarized as follows:

[1] http://framenet.icsi.berkeley.edu

[2] http://en.wikipedia.org

Step 1 For each lexical unit l, we collect from the English Wikipedia dump[3] all contexts[4] where l is the anchor of an internal link (wiki link). The set of targets represents the senses of l in Wikipedia and the contexts are used as labelled training examples. For example, the lexical unit *building.n* in the frame *Buildings* is an anchor in 708 different contexts that point to 42 different Wikipedia pages (senses).

Step 2 The set of contexts with their corresponding senses is then used to train the WSD system described in Section 4. For example, the context "The *building*, which date from the mid-to-late 19th century, were built in a variety of High Victorian architectural styles." is a training example for the sense defined by the Wikipedia page *Building*.

Step 3 Finally, the disambiguation model learned in the previous step is used to map a pair (F, l) to a Wikipedia article. (F, l) is represented as a fictitious-context derived by aggregating the frame definition and all lexical units associated to F. We used the term "fictitious-context" to remark the slight difference in structure compared with the training contexts (i.e., the Wikipedia paragraphs). For example, "... structures forming an enclosure and providing protection from the elements ... acropolis arena auditorium bar building ..." is the fictitious-context built for the pair *(Buildings, building.n)*. The sense, i.e., the Wikipedia article, assigned to the fictitious-context by the disambiguation algorithm uniquely defines the mapping. The previous example is assigned to the Wikipedia page *Building*.

4 The WSD Algorithm

Gliozzo et al. (2005) proposed an elegant approach to WSD based on kernel methods. The algorithm proved effective at Senseval-3 (Mihalcea and Edmonds, 2004) and, nowadays, it still represents the state-of-the-art in WSD (Pradhan et al., 2007). Specifically, they addressed these issues: (i) independently modeling domain and syntagmatic aspects of sense distinction to improve feature representativeness; and (ii) exploiting external knowledge acquired from unlabeled data, with the purpose of drastically reducing the amount of labeled training data. The first direction is based on the linguistic assumption that syntagmatic and domain (associative) relations are crucial for representing sense distictions, but they are originated by different phenomena. Regarding the second direction, it is possible to obtain a more accurate prediction by taking into account unlabeled data relevant for the learning problem (Chapelle et al., 2006).

On the other hand, kernel methods are theoretically well founded in statistical learning theory and shown good empirical results in many applications (Shawe-Taylor and Cristianini, 2004). The strategy adopted by kernel methods consists of splitting the learning problem into two parts. They first embed the input data in a suitable feature space, and then use a linear algorithm (e.g., support vector machines) to discover nonlinear patterns in the input space. The kernel function is the only task-specific component of the learning algorithm. Thus, to develop a WSD system, one only needs to define appropriate kernel functions to represent the domain and syntagmatic aspects of sense distinction and to exploit the properties of kernel functions in order to define a composite kernel that combines and extends individual kernels.

The WSD system described in the following consists of a composite kernel (Section 4.3) that combines the domain and syntagmatic kernels. The former (Section 4.1) models the domain aspects of sense distinction, the latter (Section 4.2) represents the syntagmatic aspects of sense distinction.

4.1 Domain Kernel

It is been shown that domain information is fundamental for WSD (Magnini et al., 2002). For instance, the (domain) polysemy between the computer science and the medicine senses of the word "virus" can be solved by considering the domain of the context in which it appears.

In the context of kernel methods, domain information can be exploited by defining a kernel function that estimates the domain similarity between the contexts of the word to be disambiguated. The simplest method to estimate the domain similarity between two texts is to compute the cosine similarity of their vector representations in the vector space model (VSM). The VSM is a k-dimensional space $\mathbb{R}^k$, in which the text t_j is represented by a vector $\vec{t_j}$, where the i^{th} component is the term

[3] http://download.wikimedia.org/enwiki/20090306

[4] A context corresponds to a line of text in the Wikipedia dump and it is represented as a paragraph in a Wikipedia article.

frequency of the term w_i in t_j. However, such an approach does not deal well with lexical variability and ambiguity. For instance, despite the fact that the sentences "he is affected by AIDS" and "HIV is a virus" express concepts closely related, their similarity is zero in the VSM because they have no words in common (they are represented by orthogonal vectors). On the other hand, due to the ambiguity of the word "virus" , the similarity between the sentences "the laptop has been infected by a virus" and "HIV is a virus" is greater than zero, even though they convey very different messages.

To overcome this problem, Gliozzo et al. (2005) introduced the domain model (DM) and show how to define a domain VSM in which texts and terms are represented in a uniform way. A DM is composed of soft clusters of terms. Each cluster represents a semantic domain, that is, a set of terms that often co-occur in texts having similar topics. A DM is represented by a $k \times k'$ rectangular matrix $\mathbf{D}$, containing the degree of association among terms and domains.

The matrix $\mathbf{D}$ is used to define a function $\mathcal{D} : \mathbb{R}^k \rightarrow \mathbb{R}^{k'}$, that maps the vector $\vec{t_j}$ represented in the standard VSM, into the vector $\vec{t'_j}$ in the domain VSM. $\mathcal{D}$ is defined by

$$\mathcal{D}(\vec{t_j}) = \vec{t_j}(\mathbf{I^{IDF}D}) = \vec{t'_j}, \tag{1}$$

where $\vec{t_j}$ is represented as a row vector, $\mathbf{I^{IDF}}$ is a $k \times k$ diagonal matrix such that $i^{IDF}_{i,i} = IDF(w_i)$, and $IDF(w_i)$ is the inverse document frequency of w_i.

In the domain space, the similarity is estimated by taking into account second order relations among terms. For example, the similarity of the two sentences "He is affected by AIDS" and "HIV is a virus" is very high, because the terms AIDS, HIV and virus are strongly associated with the domain medicine.

Singular valued decomposition (SVD) is used to acquire in a unsupervised way the DM from a corpus represented by its term-by-document matrix $\mathbf{T}$. SVD decomposes the term-by-document matrix $\mathbf{T}$ into three matrixes $\mathbf{T} \simeq \mathbf{V}\mathbf{\Sigma_{k'}}\mathbf{U}^T$, where $\mathbf{V}$ and $\mathbf{U}$ are orthogonal matrices (i.e., $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ and $\mathbf{U}^T\mathbf{U} = \mathbf{I}$) whose columns are the eigenvectors of $\mathbf{T}\mathbf{T}^T$ and $\mathbf{T}^T\mathbf{T}$ respectively, and $\mathbf{\Sigma_{k'}}$ is the diagonal $k \times k$ matrix containing the highest $k' \ll k$ eigenvalues of $\mathbf{T}$, and all the remaining elements set to 0. The parameter k' is the dimensionality of the domain VSM and can be fixed in advance. Under this setting, the domain matrix $\mathbf{D}$ is defined by

$$\mathbf{D} = \mathbf{I^N}\mathbf{V}\sqrt{\mathbf{\Sigma_{k'}}} \tag{2}$$

where $\mathbf{I^N}$ is a diagonal matrix such that $\mathbf{i^N_{i,i}} = \frac{1}{\sqrt{\langle \vec{w'_i}, \vec{w'_i} \rangle}}$, $\vec{w'_i}$ is the i^{th} row of the matrix $\mathbf{V}\sqrt{\mathbf{\Sigma_{k'}}}$. The domain kernel is explicitly defined by

$$K_D(t_i, t_j) = \langle \mathcal{D}(t_i), \mathcal{D}(t_j) \rangle, \tag{3}$$

where $\mathcal{D}$ is the domain mapping defined in Equation 1. Finally, the domain kernel is further extended to include the standard bag-of-word kernel.

4.2 Syntagmatic Kernel

Kernel functions are not restricted to operate on vectorial objects $\vec{x} \in \mathbb{R}^k$. In principle, kernels can be defined for any kind of object representation, such as strings and trees. As syntagmatic relations hold among words collocated in a particular temporal order, they can be modeled by analyzing sequences of words. Therefore, the string kernel (Shawe-Taylor and Cristianini, 2004) is a valid tool to represent such relations. It counts how many times a (non-contiguous) subsequence of symbols u of length n occurs in the input string s, and penalizes non-contiguous occurrences according to the number of gaps they contain. Formally, let V be the vocabulary, the feature space associated with the string kernel of length n is indexed by a set I of subsequences over V of length n. The (explicit) mapping function is defined by

$$\phi^n_u(s) = \sum_{\mathbf{i}:u=s(\mathbf{i})} \lambda^{l(\mathbf{i})}, u \in V^n, \tag{4}$$

where $u = s(\mathbf{i})$ is a subsequence of s in the positions given by the tuple $\mathbf{i}$, $l(\mathbf{i})$ is the length spanned by u, and $\lambda \in]0, 1]$ is the decay factor used to penalize non-contiguous subsequences.

The associated string kernel is defined by

$$K_n(s_i, s_j) = \langle \phi^n(s_i), \phi^n(s_j) \rangle = \sum_{u \in V^n} \phi^n(s_i)\phi^n(s_j) \tag{5}$$

Gliozzo et al. (2005) modified the generic definition of the string kernel in order to take into account (sparse) collocations. Specifically, they defined syntagmatic kernels as a combination of string kernels applied to sequences of words in a fixed-size window centered on the word to be disambiguated. This formulation allows estimating the number of common (sparse) subsequences of

words (i.e., collocations) between two examples, in order to capture syntagmatic similarity. The syntagmatic kernel is defined by

$$K_S(s_i, s_j) = \sum_{n=1}^{p} K_n(s_i, s_j), \quad (6)$$

where K_n is the string kernel defined in Equation 5 and the parameter n represents the length of the subsequences analyzed when estimating the similarity between contexts. Notice that the syntagmatic kernel is only effective for those fictitious contexts in which the lexical units do occur in meaningful sentences, however this is not guaranteed for the lexical units without examples.

4.3 Composite Kernel

Finally, to combine domain and syntagmatic information, the composite kernel is defined by

$$K_{WSD}(t_i, t_j) = \hat{K}_D(t_i, t_j) + \hat{K}_S(t_i, t_j), \quad (7)$$

where $\hat{K}_D$ and $\hat{K}_S$ are normalized kernels defined in Equation 3 and 6, respectively.[5] It follows directly from the explicit construction of the feature space and from closure properties of kernels that it is a valid kernel.

5 Mapping task

In this section we report the first experiment, namely the mapping between (F, l) pairs and a Wikipedia pages. We describe the experimental setup and then present the corresponding evaluation.

5.1 Experimental setup

We applied our algorithm to all lexical units that do not have any example sentence in the FrameNet database. In principle, the proposed approach can be applied to every lexical unit, and we expect the algorithm performance to improve if some example sentences are already available because they could be added to the fictitious-context used to represent (F, l) in the system. Nevertheless, in this explorative study we wanted to focus on the harder cases, even if results are likely to be worse than on the whole FrameNet database.

In FrameNet, 3,305 (F, l) pairs have no example sentences (536 pairs with adjectival LU, 1313 verbal LU, 1456 nominal LU). Since Wikipedia is basically a resource organized by concepts, which are generally expressed by nouns, we decided to restrict our experiment to nominal lexical units. Besides, many verbal and adjectival concepts in Wikipedia are redirected to nominal identifiers. So, we randomly selected 900 pairs with nominal lexical units. For the moment, we decided to discard lexical units expressed by multiwords (about 150), which will be taken into account in a future version of our system. The average ambiguity of the 900 LUs considered is 1.24 in FrameNet. Instead, every LU corresponds to about 35 candidate senses in Wikipedia.

In order to perform WSD, we built the domain model from the 200,000 most visited Wikipedia articles. After removing terms that occur less than 5 times, the resulting dictionaries contain about 300,000 terms. We used the SVDLIBC package[6] to compute the SVD, truncated to 100 dimensions. The experiments were performed using the SVM package *LIBSVM* (Chang and Lin, 2001) customized to embed the kernels described in Section 4.

5.2 Evaluation

In this first evaluation step, we focus on the quality of the mapping between (F, l) pairs and Wikipedia articles. In order to evaluate the system output, we created a gold standard where 250 (F, l) pairs randomly extracted from the nominal subset described above have been manually linked to the Wikipedia page (if available) that best corresponds to the meaning of l in F. The pairs have been chosen in order to maximize the frame variability, i.e. every pair corresponds to a different frame. Since our gold standard contains 34% of all frames in the FrameNet database, we believe that, despite its limited size, it is well representative of FrameNet characteristics. Evaluation was carried out comparing the system output against the gold standard. Results are reported in Table 1. The baseline was computed considering the *most frequent sense* of every lexical unit in Wikipedia. This element is obtained by taking into account all occurrences in Wikipedia where the lexical unit LU we consider is anchored to a given page. The most frequent sense for LU is the page to which LU is most frequently linked in Wikipedia. Since about 14% of the lexical units in the gold standard are not present in Wikipedia, we also estimated an upper bound accuracy of 0.86. This confirms our in-

[5] $\hat{K}(x_i, x_j) = \frac{K(x_i, x_j)}{\sqrt{K(x_j, x_j)K(x_i, x_i)}}$

[6] http://tedlab.mit.edu/~dr/svdlibc/

tuition that FrameNet and Wikipedia are linkable resources to a large extent and that our task is well-founded.

	Accuracy
Baseline	0.66
System output	**0.71**
Upper bound	0.86

Table 1: Accuracy evaluation.

Wrong assignments include also problematic cases that are not directly connected to proper system errors. One of the most relevant issues is the different granularity between FrameNet frames and Wikipages. For example, the NETWORK frame is defined as *"a set of entities of the same or similar types (Nodes) are linked to each other by Connections to form a Network allowing for the flow of information, resources, etc."*. Even if the listed lexical units (*network.n* and *web.n*) and some examples refer to the informatics domain, the situation described in the FrameNet database is more general. Wikipedia instead lists several pages that may be seen as subdomains of NETWORK such as *Computer_network*, *Social_network*, *Telecommunications_network*, etc. In the future, it may be worth modifying the system in order to allow multiple assignments of Wikipages for every frame.

In other cases, frame definitions seem not to be very consistent and it is very difficult to discriminate between two frames even for a human annotator. For example, ESTIMATED_VALUE and ESTIMATING include both *estimation.n* as lexical unit, but since their frame definitions are almost the same and the other lexical units in the same frame are not discriminative, the system links both (F, l) pairs to the same Wikipedia article.

6 English FrameNet expansion

In the following part of the experiment, we want to investigate to what extent the FrameNet – Wikipedia mapping can be effectively applied to automatically expand the FrameNet database with new example sentences, and eventually to acquire new lexical units. For every (F, l) pair, we consider the linked Wikipedia sense *s* and extract all sentences C_s in Wikipedia with a reference to s. In this way, we can assume that, if s was linked to (F, l), C_s can be included in the example sentences of F. This repository of sentences is already divided by sense and can significantly speed-up manual annotation. On the other hand, the extracted sentences could enrich the training set of machine learning systems for frame annotation to improve the frame identification step. In fact, this task has raised growing interest in the NLP community, with a devoted challenge at the last SemEval campaign (Baker et al., 2007).

This retrieval process allows also to extract from C_s all words W_s that have an embedded reference to s in the form *word*. In this way, W_s are automatically included in F as new lexical units. In this phase, redirecting links are very useful because they automatically connect a word or expression to its nearest sense in case there is no specific page for this word. The information about redirecting allows also to account for orthographic variations of the same lexical unit, for example *collectible* is redirected to *collectable*.

We explain the data extraction process in the light of an example from our dataset. Our WSD system assigned to the (F, l) pair (WORD_RELATIONS – *homonym.n*) the Wikipage `http://en.wikipedia.org/wiki/Homonym`. So, we extracted from the English Wikipedia dump all sentences where the anchor *<a href="/wiki/Homonym"...* > appears and assumed that the word or multiword expression that is linked to the *Homonym* site may be a good candidate as lexical unit for the WORD_RELATIONS frame. In this case, the example sentences were 186. Apart from *homonym*, the candidate lexical units are *homograph*, *homophone*, *homophonous*, *homonymic*, *heteronym*, *same*. Among them, only the latter is not appropriate, even if the sentence where it occurs is semantically connected to the WORD_RELATIONS frame: "In Hebrew the word 'thus' has the *same* triconsonantal root". Instead, *homonymic* and *heteronym* can be acquired as new lexical units for WORD_RELATIONS, and *homograph*, for which no example sentences are provided in FrameNet, can be automatically instantiated by a set of examples.

6.1 Experimental setup

We considered 893 frame – lexical unit pairs assigned to Wikipedia pages following the algorithm described in Section 3. We discarded 7 pairs for which the system reported an assignment failure, i.e. the best sense delivered is the disambigua-

tion page. Then we extracted a set of sentences for every (F, l) pair as described in the previous paragraph. Statistics about the retrieved data is reported in Table 2.

	English Wikipedia
(F, l) pairs	893
N. of extracted sents	964,268
Avg. sents per (F, l)	1,080

Table 2: Extracted data from English Wikipedia

6.2 Evaluation

The dimension of the extracted corpus does not allow to carry out a comprehensive evaluation. For this reason, we manually evaluated 1,000 sentences, i.e. we considered 20 (F, l) pairs, and for each of them we evaluated 50 sentences extracted from our large repository. Both (F, l) pairs and the assigned sentences were randomly selected. In particular, the 20 (F, l) pairs do not contain only correctly assigned pairs, in fact three of them are wrong. Anyhow, the 20 pairs seem to be a representative subset of the 893 pairs considered in our experiment because they include both monosemic lexical units (*gynaecology.n* in MEDICAL_SPECIALTIES) and more ambiguous ones (*club.n* in the WEAPON frame).

Our evaluation shows that 78% of the sentences were correctly linked to (F, l) pairs. This value is higher that the mapping accuracy between (F, l) and Wikipages reported in Section 5.2. In fact, we noticed that even if the Wikipage assigned to (F, l) is not the article that best corresponds to the meaning of l in F, some sentences pointing to it may be appropriate to express l.

As we already mentioned in Section 5.2, the different granularity of the information encoded by frames and Wikipages impacts on the output quality. For example, *conversion.n* in CAUSE_CHANGE has a causative meaning, while it implies a personal process in UNDERGO_CHANGE. The mapping, instead, links (CAUSE_CHANGE – *conversion.n*) to the *Religious_conversion* page, and all the sentences collected point to religious conversion, regardless of their causative form or not. Another characteristic of this approach is that we can acquire new lexical units regardless of their part-of-speech, even if we start from nominal lexical units. This proves that we do not need to apply the initial mapping to verbal or adjectival LUs to obtain new data for all parts of speech. For example, we linked (MEDICAL_SPECIALTIES – *gynaecology.n*) to the *Gynaecology* Wikipage. Consequently, we could include the adjective *gynaecologic*, pointing to the *Gynaecology* page, into the MEDICAL_SPECIALTIES frame for sentences like "Fellowship training in a *gynaecologic* subspeciality can range from one to four years". However, this advantage can also turn into a weakness, because *gynaecologist* is also redirected to the *Gynaecology* page, but it belongs to MEDICAL_PROFESSIONALS and should not be included into MEDICAL_SPECIALTIES.

For the 20 (F, l) pairs considered in the given sentences, it was possible also to retrieve 8 lexical units that are not present in FrameNet, for example *billy-club* for the WEAPON frame. Exploiting redirections and anchoring strategies, our induction method can account for orthographical variations, for example it acquires both *memorize* and *memorise*. On the other hand, also misspelled words may be collected, for instance *gynaecologial* instead of *gynaecological*.

7 Multilingual FrameNet expansion

One of the great advantages of Wikipedia is its availability in several languages. The English version is by far the most extended, but a considerable repository of pages is available also for other languages, esp. European ones. In general, articles on the same object in different languages are edited independently and do not have to be translations of one another, but are linked to each other by their authors. In this way, the multilingual versions of Wikipedia can be easily exploited to build comparable corpora, with connected Wikipages in different languages dealing with the same contents.

In this research step, we focus on this aspect of Wikipedia and propose a methodology that, using the English Wikipages as a bridge, automatically acquires new lexical units and example sentences also for other languages. This would represent the starting point towards the creation of FrameNet for new languages. Indeed, FrameNet structure comprises a language-independent level of information, namely frame and frame element definitions, and a language-dependent one, i.e. the lexical units and the example sentences. This makes the resource particularly suitable to corpus-based (semi) automatic creation of FrameNet for new languages, because the descriptive part can be pre-

served and the language-dependent layer can be populated with new instances in other languages (Crespo and Buitelaar, 2008).

We apply our extraction algorithm to the Italian Wikipedia. Since several approaches have been experimented to (semi) automatically build Italian FrameNet using WordNet (De Cao et al. (2008) and Tonelli and Pighin (2009)), we believe that our new proposal to exploit Wikipedia may be of interest in the research community. Anyhow, the approach can be exploited in principle for every language available in Wikipedia.

7.1 Experimental setup

Similarly to the data extraction process described in Section 6, we consider for every (F, l) pair in English the linked Wikipedia sense s, in English as well. Then, we retrieve the Italian Wikipedia sense s_i linked to s and extract all sentences C_i in the Italian Wikipedia dump[7] with a reference to s_i. In this way, we can assume that C_i are example sentences of F and that the words or expressions W_i in C_i containing an embedded reference to s_i are good candidate lexical units of F in the Italian FrameNet. For example, if we link `http://en.wikipedia.org/wiki/Court` to the JUDICIAL_BODY frame, we first retrieve the Italian version of the site `http://it.wikipedia.org/wiki/Tribunale`. Then, with a top-down strategy, we further extract all Italian sentences pointing to the `Tribunale` page and acquire as lexical units all words with an embedded reference to this concept, for example *tribunale* and *corte*. In this way, we can include the extracted lexical units and the sentences where they occur in the JUDICIAL_BODY frame for Italian.

Given the 893 (F, l) pairs in English and the linked Wikipedia senses described in 6.2, we first extracted the Italian Wikipages that are linked to the English ones. Then for every linked Wikipage in Italian, we retrieved all sentences with a reference pointing to that page in the Italian Wikipedia dump. Statistics about the extracted data are reported in Table 3.

Since the Italian Wikipedia is about one fifth of the English one, it was not possible to map every English Wikipage with an Italian article. In fact, only 371 senses out of 893 in English were linked to an Italian page. Also the average number of sentences extracted for every sense is much smaller (62 vs. 1,080). Anyhow, this does not represent a problem because in the English FrameNet, the lexical units whose annotation is considered to be *complete* are usually instantiated by set of 20 annotated sentences on average. So, according to the FrameNet standard, 60 sentences are more than enough to represent the meaning of a lexical unit in a frame.

	Italian Wikipedia
Linked Wikipages in Italian	371
N. of extracted sents	23,078
Avg. sents per Italian sense	62

Table 3: Extracted data from Italian Wikipedia

7.2 Evaluation

In this evaluation part, we took into account 1,000 sentences, in order to have a comparable dataset w.r.t. the evaluation for English. However, the sets of Italian sentences extracted for every (F, l), i.e. for every Wikipedia article, were much smaller, so we increased the number of randomly chosen (F, l) pairs to 80. Our evaluation is focused on the quality of the sentences and aims at assessing if the given sentences are correctly assigned to the (F, l) pairs. We report 69% accuracy, which is 9% lower than for English. Apart from the same errors and issues reported for English, a decrease in performance can be explained by the fact that, since less articles are present w.r.t. the English version, redirections and internal links tend to be less precise and fine-grained. For example, the word "*diritti*" in the sense of "*(human) rights*" redirects to the article about *Diritto*, corresponding to *Law* as a system of rules. On the contrary, *Law* and *Rights* have two different pages in English. Besides, the different quality of the two resources can also depend on the smaller number of users that edit and check the Italian articles. From the 1,000 sentences evaluated we extracted 145 new lexical units: since Italian FrameNet does not exist yet, every lexical unit in a sentence that is correct can be straightforwardly included in the first version of the resource.

8 Conclusions and Future work

In this work, we have proposed to apply a word sense disambiguation system to a new task, namely the linking between FrameNet and Wikipedia. Results are promising and show that

[7] `http://download.wikimedia.org/itwiki/20090203`

the task is adequately substantiated. The proposed approach can help enriching FrameNet with new example sentences and lexical units and provide a starting point for the creation of FrameNet-like resources in all Wikipedia languages. On the one hand, the retrieved data could speed up human annotation, requiring only a manual validation. On the other hand, the extracted sentences could provide enough training data to machine learning systems for frame assignment, since insufficient frame attestations in the FrameNet database are a major problem for such systems.

In the next research step, we plan to carry out an extended evaluation process in order to compute inter-annotator agreement and eventually point out validation problems. Then, we want to extend the mapping and the data extraction process to all (F, l) pairs in FrameNet (about 10,000). The retrieved sentences will be made available as training or annotation material. Besides, we want to create an online resource where the links between (F, l) pairs and Wikipages are made explicit and where users can browse the retrieved sentences. The resource can be produced and made available with a reduced effort for every language in Wikipedia. Anyway, the English version has proved to be more precise, while the resource for new languages would require a more accurate revision.

Acknowledgments

Claudio Giuliano is supported by the ITCH project (`http://itch.fbk.eu`), sponsored by the Italian Ministry of University and Research and by the Autonomous Province of Trento and the X-Media project (`http://www.x-media-project.org`), sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978.

References

Collin F. Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval-2007 Task 10: Frame Semantic Structure Extraction. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 99–104, Prague, CZ, June.

Aljoscha Burchardt and Anette Frank. 2006. Approximating Textual Entailment with LFG and FrameNet Frames. In *Proceedings of the 2nd PASCAL RTE Workshop*, pages 92–97, Venice, Italy.

Diego De Cao, Danilo Croce, Marco Pennacchiotti, and Roberto Basili. 2008. Combining Word Sense and Usage for modeling Frame Semantics. In *Proceedings of STEP 2008*, Venice, Italy.

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

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. 2006. *Semi-Supervised Learning*. MIT Press, Cambridge, MA.

Ian Chow and Jonathan Webster. 2007. Integration of Linguistic Resources for Verb Classification: FrameNet Frame, WordNet Verb and Suggested Upper Merged Ontology. *Computational Linguistics and Intelligent Text Processing*, pages 1–11.

Mario Crespo and Paul Buitelaar. 2008. Domain-specific English-to-Spanish Translation of FrameNet. In *Proc. of LREC 2008*, Marrakech.

Andras Csomai and Rada Mihalcea. 2008. Linking Documents to Encyclopedic Knowledge. *IEEE Intelligent Systems, special issue on "Natural Language Processing for the Web"*.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June. Association for Computational Linguistics.

Katrin Erk. 2004. Frame assignment as Word Sense Disambiguation. In *Proceedings of IWCS-6*, Tilburg, NL.

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

C.J. Fillmore, C.R. Johnson, and M. R. L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16:235–250, September.

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

A. Gliozzo, C. Giuliano, and C. Strapparava. 2005. Domain kernels for word sense disambiguation. In *Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL-05)*, pages 403–410, Ann Arbor, Michigan, June.

R. Johansson and P. Nugues. 2007. Using WordNet to extend FrameNet coverage. In *Proc. of the Workshop on Building Frame-semantic Resources for Scandinavian and Baltic Languages, at NODALIDA*, Tartu.

B. Magnini, C. Strapparava, G. Pezzulo, and A. Gliozzo. 2002. The Role of Domain Information in Word Sense Disambiguation. *Natural Language Engineering*, 8(4):359–373.

R. Mihalcea and P. Edmonds, editors. 2004. *Proceedings of SENSEVAL-3*, Barcelona, Spain, July.

David Milne and Ian H. Witten. 2008. Learning to link with Wikipedia. In *CIKM '08: Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518, NY, USA. ACM.

Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. Semeval-2007 Task-17: English Lexical Sample, SRL and All Words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 87–92, Prague, Czech Republic, June. Association for Computational Linguistics.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Lei Shi and Rada Mihalcea. 2004. Open Text Semantic Parsing Using FrameNet and WordNet. In *Proceedings of HLT-NAACL 2004*.

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

Sara Tonelli and Daniele Pighin. 2009. New features for FrameNet - WordNet Mapping. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, Boulder, CO, USA.

Fast, Cheap, and Creative: Evaluating Translation Quality Using Amazon's Mechanical Turk

Chris Callison-Burch
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, Maryland
ccb@cs.jhu.edu

Abstract

Manual evaluation of translation quality is generally thought to be excessively time consuming and expensive. We explore a fast and inexpensive way of doing it using Amazon's Mechanical Turk to pay small sums to a large number of non-expert annotators. For $10 we redundantly recreate judgments from a WMT08 translation task. We find that when combined non-expert judgments have a high-level of agreement with the existing gold-standard judgments of machine translation quality, and correlate more strongly with expert judgments than Bleu does. We go on to show that Mechanical Turk can be used to calculate human-mediated translation edit rate (HTER), to conduct reading comprehension experiments with machine translation, and to create high quality reference translations.

1 Introduction

Conventional wisdom holds that manual evaluation of machine translation is too time-consuming and expensive to conduct. Instead, researchers routinely use automatic metrics like Bleu (Papineni et al., 2002) as the sole evidence of improvement to translation quality. Automatic metrics have been criticized for a variety of reasons (Babych and Hartley, 2004; Callison-Burch et al., 2006; Chiang et al., 2008), and it is clear that they only loosely approximate human judgments. Therefore, having people evaluate translation output would be preferable, if it were more practical.

In this paper we demonstrate that the manual evaluation of translation quality is not as expensive or as time consuming as generally thought. We use Amazon's Mechanical Turk, an online labor market that is designed to pay people small sums of money to complete *human intelligence tests* – tasks that are difficult for computers but easy for people. We show that:

- Non-expert annotators produce judgments that are very similar to experts and that have a stronger correlation than Bleu.
- Mechanical Turk can be used for complex tasks like human-mediated translation edit rate (HTER) and creating multiple reference translations.
- Evaluating translation quality through reading comprehension, which is rarely done, can be easily accomplished through creative use of Mechanical Turk.

2 Related work

Snow et al. (2008) examined the accuracy of labels created using Mechanical Turk for a variety of natural language processing tasks. These tasks included word sense disambiguation, word similarity, textual entailment, and temporal ordering of events, but not machine translation. Snow et al. measured the quality of non-expert annotations by comparing them against labels that had been previously created by expert annotators. They report inter-annotator agreement between expert and non-expert annotators, and show that the average of many non-experts converges on performance of a single expert for many of their tasks.

Although it is not common for manual evaluation results to be reported in conference papers, several large-scale manual evaluations of machine translation quality take place annually. These include public forums like the NIST MT Evaluation Workshop, IWSLT and WMT, as well as the project-specific Go/No Go evaluations for the DARPA GALE program. Various types of human judgments are used. NIST collects 5-point fluency and adequacy scores (LDC, 2005), IWSLT and

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 286–295,
Singapore, 6-7 August 2009.

WMT collect relative rankings (Callison-Burch et al., 2008; Paul, 2006), and DARPA evaluates using HTER (Snover et al., 2006). The details of these are provided later in the paper. Public evaluation campaigns provide a ready source of gold-standard data that non-expert annotations can be compared to.

3 Mechanical Turk

Amazon describes its Mechanical Turk web service[1] as *artificial* artificial intelligence. The name and tag line refer to a historical hoax from the 18th century where an automaton appeared to be able to beat human opponents at chess using a clockwork mechanism, but was, in fact, controlled by a person hiding inside the machine. The Mechanical Turk web site provides a way to pay people small amounts of money to perform tasks that are simple for humans but difficult for computers. Examples of these Human Intelligence Tasks (or HITs) range from labeling images to moderating blog comments to providing feedback on relevance of results for a search query.

Anyone with an Amazon account can either submit HITs or work on HITs that were submitted by others. Workers are sometimes referred to as "Turkers" and people designing the HITs are "Requesters." Requesters can specify the amount that they will pay for each item that is completed. Payments are frequently as low as $0.01. Turkers are free to select whichever HITs interest them.

Amazon provides three mechanisms to help ensure quality: First, Requesters can have each HIT be completed by multiple Turkers, which allows higher quality labels to be selected, for instance, by taking the majority label. Second, the Requester can require that all workers meet a particular set of qualications, such as sufficient accuracy on a small test set or a minimum percentage of previously accepted submissions. Finally, the Requester has the option of rejecting the work of individual workers, in which case they are not paid.

The level of good-faith participation by Turkers is surprisingly high, given the generally small nature of the payment.[2] For complex undertakings like creating data for NLP tasks, Turkers do not have a specialized background in the subject, so there is an obvious tradeoff between hiring individuals from this non-expert labor pool and seeking out annotators who have a particular expertise.

[1] http://www.mturk.com/

[2] For an analysis of the demographics of Turkers and why they participate, see: http://behind-the-enemy-lines.blogspot.com/2008/03/mechanical-turk-demographics.html

4 Experts versus non-experts

We use Mechanical Turk as an inexpensive way of evaluating machine translation. In this section, we measure the level of agreement between expert and non-expert judgments of translation quality. To do so, we recreate an existing set of gold-standard judgments of machine translation quality taken from the Workshop on Statistical Machine Translation (WMT), which conducts an annual large-scale human evaluation of machine translation quality. The experts who produced the gold-standard judgments are computational linguists who develop machine translation systems.

We recreated all judgments from the WMT08 German-English News translation task. The output of the 11 different machine translation systems that participated in this task was scored by ranking translated sentences relative to each other. To collect judgements, we reproduced the WMT08 web interface in Mechanical Turk and provided these instructions:

Evaluate machine translation quality *Rank each translation from Best to Worst relative to the other choices (ties are allowed). If you do not know the source language then you can read the reference translation, which was created by a professional human translator.*

The web interface displaced 5 different machine translations of the same source sentence, and had radio buttons to rate them.

Turkers were paid a grand total of $9.75 to complete nearly 1,000 HITs. These HITs exactly replicated the 200 screens worth of expert judgments that were collected for the WMT08 German-English News translation task, with each screen being completed by five different Turkers. The Turkers were shown a source sentence, a reference translation, and translations from five MT systems. They were asked to rank the translations relative to each other, assigning scores from best to worst and allowing ties.

We evaluate non-expert Turker judges by measuring their inter-annotator agreement with the WMT08 expert judges, and by comparing the correlation coefficient across the rankings of the machine translation systems produced by the two sets of judges.

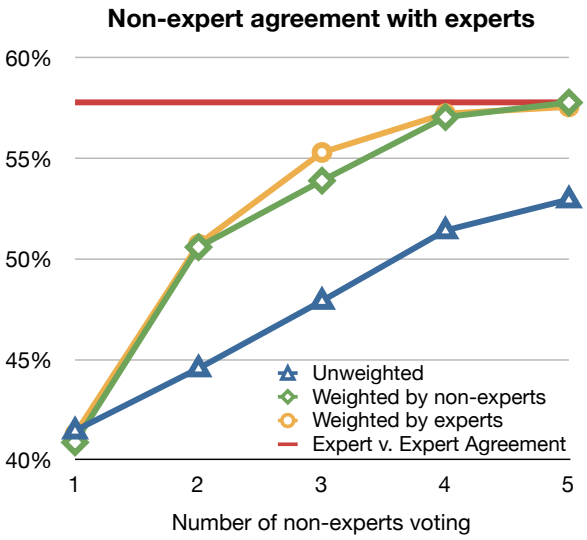


Figure 1: Agreement on ranking translated sentences increases as more non-experts vote. Weighting non-experts' votes based on agreement with either experts or other non-expert increases it up further. Five weighted non-experts reach the top line agreement between experts.

Combining ranked judgments Each item is redundantly judged by five non-experts. We would like to combine of their judgments into a single judgment. Combining ranked judgments it is more complicated than taking simple majority vote. We use techniques from *preference voting*, in which voters rank a group of candidates in order of preference. To create an ordering from the the ranks assigned to the systems by multiple Turkers, we use Schulze's method (Schulze, 2003). It is guaranteed to correctly pick the winner that is preferred pairwise over the other candidates. It further allows a complete ranking of candidates to be constructed, making it a suitable method for combining ranked judgments.

Figure 1 shows the effect of combining non-experts judgments on their agreement with experts. Agreement is measured by examining each pair of translated sentence and counting when two annotators both indicated that A $>$ B, A $<$ B, or A $=$ B. Chance agreement is $\frac{1}{3}$. The top line indicates the inter-annotator agreement between WMT08 expert annotators, who agreed with each other 58% of the time. When we have only a single non-expert annotator's judgment for each item, the agreement with experts is only 41%. As we increase the number of non-experts to five, their agreement with experts improves to 53%, if their votes are counted equally.

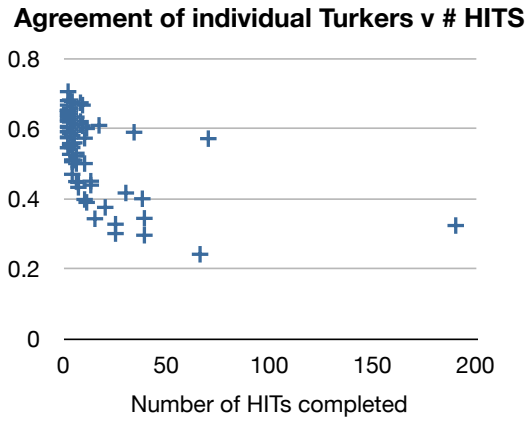


Figure 2: The agreement of individual Turkers with the experts. The most prolific Turker performed barely above chance, indicating random clicking. This suggests that users who contribute more tend to have lower quality.

Weighting votes Not all Turkers are created equal. The quality of their works varies. Figure 2 shows the agreement of individual Turkers with expert annotators, plotted against the number of HITs they completed. The figure shows that their agreement varies considerably, and that Turker who completed the most judgments was among the worst performing.

To avoid letting careless annotators drag down results, we experimented with weighted voting. We weighted votes in two ways:

- Votes were weighted by measuring agreement with experts on the 10 initial judgments made. This would be equivalent to giving Turkers a pretest on gold standard data and then calibrating their contribution based on how well they performed.
- Votes were weighted based on how often one Turker agreed with the rest of the Turkers over the whole data set. This does not require any gold standard calibration data. It goes beyond simple voting, because it looks at a Turker's performance over the entire set, rather than on an item-by-item basis.

Figure 1 shows that these weighting mechanisms perform similarly well. For this task, deriving weights from agreement with other non-experts is as effective as deriving weights from experts. Moreover, by weighting the votes of five Turkers,

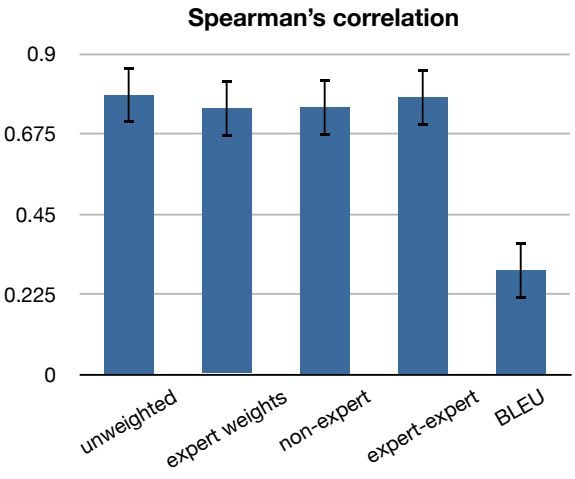


Figure 3: Correlation with experts' ranking of systems. All of the different ways of combining the non-expert judgments perform at the upper bound of expert-expert correlation. All correlate more strongly than Bleu.

we are able to achieve the same rate of agreement with experts as they achieve with each other.

Correlation when ranking systems In addition to measuring agreement with experts at the sentence-level, we also compare non-expert *system-level* rankings with experts. Following Callison-Burch et al. (2008), we assigned a score to each of the 11 MT systems based on how often its translations were judged to be better than or equal to any other system. These scores were used to rank systems and we measured Spearman's ρ against the system-level ranking produced by experts.

Figure 3 shows how well the non-expert rankings correlate with expert rankings. An upper bound is indicated by the *expert-expert* bar. This was created using a five-fold cross validation where we used 20% of the expert judgments to rank the systems and measured the correlation against the rankings produced by the other 80% of the judgments. This gave a ρ of 0.78. All ways of combining the non-expert judgments resulted in nearly identical correlation, and all produced correlation within the range of with what we would experts to.

The rankings produced using Mechanical Turk had a much stronger correlation with the WMT08 expert rankings than the Blue score did. It should be noted that the WMT08 data set does not have multiple reference translations. If multiple references were used that Bleu would likely have stronger correlation. However, it is clear that the cost of hiring professional translators to create multiple references for the 2000 sentence test set would be much greater than the $10 cost of collecting manual judgments on Mechanical Turk.

5 Feasibility of more complex evaluations

In this section we report on a number of creative uses of Mechanical Turk to do more sophisticated tasks. We give evidence that Turkers can create high quality translations for some languages, which would make creating multiple reference translations for Bleu less costly than using professional translators. We report on experiments evaluating translation quality with HTER and with reading comprehension tests.

5.1 Creating multiple reference translations

In addition to evaluating machine translation quality, we also investigated the possibility of using Mechanical Turk to create additional reference translations for use with automatic metrics like Bleu. Before trying this, we were skeptical that Turkers would have sufficient language skills to produce translations. Our translation HIT had the following instructions:

Translate these sentences *Your task is to translate 10 sentences into English. Please make sure that your English translation:*

- *Is faithful to the original in both meaning and style*
- *Is grammatical, fluent, and natural-sounding English*
- *Does not add or delete information from the original text*
- *Does not contain any spelling errors*

When creating your translation, please:

- *Do not use any machine translation systems*
- *You may look up a word on wordreference.com if you do not know its translation*

Afterwards, we'll ask you a few quick questions about your language abilities.

We solicited translations for 50 sentences in French, German, Spanish, Chinese and Urdu, and designed the HIT so that five Turkers would translate each sentence.

Filtering machine translation Upon inspecting the Turker's translations it became clear that many had ignored the instructions, and had simply cut-and-paste machine translation rather then translating the text themselves. We therefore set up a second HIT to filter these out. After receiving the

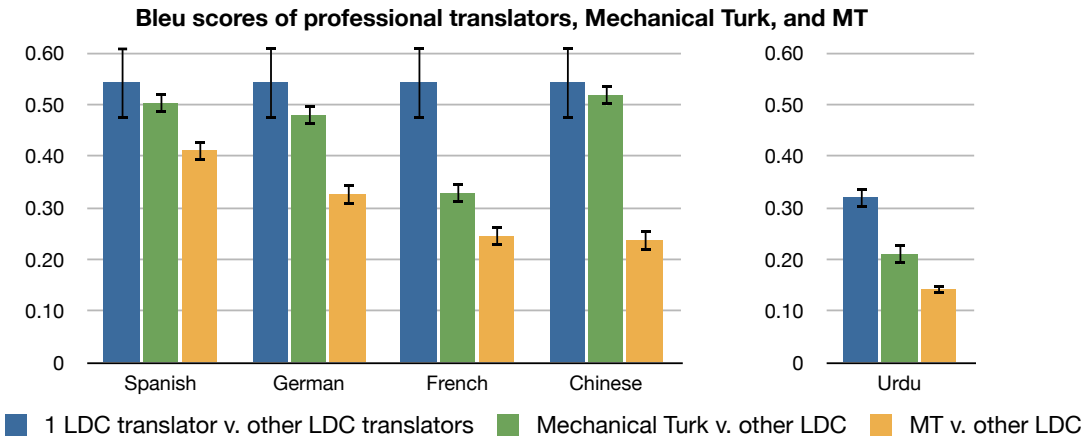


Figure 4: Bleu scores quantifying the quality of Turkers' translations. The chart shows the average Bleu score when one LDC translator is compared against the other 10 translators (or the other 2 translators in the case of Urdu). This gives an upper bound on the expected quality. The Turkers' translation quality falls within a standard deviation of LDC translators for Spanish, German and Chinese. For all languages, Turkers produce significantly better translations than an online machine translation system.

translations, we had a second group of Turkers clean the results.

Detect machine translation *Please use two online machine translation systems to translate the text into English, and then copy-and-paste the translations into the boxes below. Finally, look at a list of translations below and click on the ones that look like they came from the online translation services.*

We automatically excluded Turkers whose translations were flagged 30% of the time or more.

Quality of Turkers' translations Our 50 sentence test sets were selected so that we could compare the translations created by Turkers to translations commissioned by the Linguistics Data Consortium. For the Chinese, French, Spanish, and German translations we used the the Multiple-Translation Chinese Corpus.[3] This corpus has 11 reference human translations for each Chinese source sentence. We had bilingual graduate students translate the first 50 English sentences of that corpus into French, German and Spanish, so that we could re-use the multiple English reference translations. The Urdu sentences were taken from the NIST MT Eval 2008 Urdu-English Test Set[4] which includes three distinct English translations for every Urdu source sentence.

Figure 4 shows the Turker's translation quality in terms of the Bleu metric. To establish an upper bound on expected quality, we determined what the Bleu score would be for a professional translator when measured against other professionals. We calculated a Bleu score for each of the 11 LDC translators using the other 10 translators as the reference set. The average Bleu score for LDC2002T01 was 0.54, with a standard deviation of 0.07. The average Bleu for the Urdu test set is lower because it has fewer reference translations.

To measure the Turkers' translation quality, we randomly selected translations of each sentence from Turkers who passed the Detect MT HIT, and compared them against the same sets of 10 reference translations that the LDC translators were compared against. We randomly sampled the Turkers 10 times, and calculated averages and standard deviations for each source language. Figure 4 the Bleu scores for the Turkers' translations of Spanish, German and Chinese are within the range of the LDC translators. For all languages, the quality is significantly higher than an online machine translation system. We used Yahoo's Babelfish for Spanish, German, French and Chinese,[5] was likely and Babylon for Urdu.

Demographics We collected demographic information about the Turkers who completed the translation task. We asked how long they had spoken the source language, how long they had spo-

[3]LDC catalog number LDC2002T01

[4]LDC catalog number LDC2009E11

[5]We also compared against Google Translate, but excluded the results since its average Bleu score was better than the LDC translators, likely because the test data was used to train Google's statistical system.

Spanish		
Native lang	English (7 people), Spanish (2), English-Spanish bilingual, Portuguese	English, Hindi
Country	USA (7 people), Mexico (3), Brazil,	USA (2)
Spanish level	30+ years (2 people), 15 years (2), 6 years, 2 years (2), whole life (4)	18 years, 4 years
English level	15 years (3), whole life (9)	whole life , 15 years
German		
Native lang	German (3), Turkish (2), Italian, Danish, English, Norwegian, Hindi	Marathi, Tamil, Hindi, English
Country	Germany (3), USA, Italy, China, Denmark, Turkey, Norway, India	USA (2), India (2)
German level	20 years (2), 10 years (3), 5 years (2), 2 years, whole life (3)	10 years, 1 year (2)
English level	20+ years (4), 10-20 years (5) whole life	whole life (2), 15-20 years (2)
French		
Native lang	English (9 people), Portuguese, Hindi	English (2)
Country	USA (6), Israel, Singapore, UK, Brazil, India	USA (2)
French level	20+ years (4 people), 8-12 years (4), 5 years (2), 2 years	10 years, 1 years, 6 years
English level	whole life (9), 20 years, 15 years	whole life (2),
Chinese		
Native lang	Hindi (2)	English (3) Hindi, Marathi, Tamil
Country	India (2)	India (3), USA (3)
Chinese level	2 years, 1 year	3 years, 2 years, none
English level	18 years, 20+ years	16 years, whole life (2)
Urdu		
Native lang	Urdu (6 people)	Tamil (2), Hindi, Telugu
Country	Pakistan (3), Bahrain, India, Saudi Arabia	India (4)
Urdu level	whole life (6 people)	2 years, 1 year, never (2)
English level	20+ years (5), 15 years (2), 10 years	10+ years (5), 5 years

Table 1: Self-reported demographic information from Turkers who completed the translation HIT. The statistics on the left are for people who appeared to do the task honestly. The statistics on the right are for people who appeared to be using MT (marked as using it 20% or more in the Detect MT HIT).

ken English, what their native language was, and where they lived. Table 1 gives their replies.

Cost and speed We paid Turkers $0.10 to translate each sentence, and $0.006 to detect whether a sentence was machine translated. The cost is low enough that we could create a multiple reference set quite cheaply; it would cost less than $1,000 to create 4 reference translations for 2000 sentences.

The time it took for the 250 translations to be completed for each language varied. It took less than 4 hours for Spanish, 20 hours for French, 22.5 hours for German, 2 days for Chinese, and nearly 4 days for Urdu.

5.2 HTER

Human-mediated translation edit rate (HTER) is the official evaluation metric of the DARPA GALE program. The evaluation is conducted annually by the Linguistics Data Consortium, and it is used to determine whether the teams participating the program have met that year's benchmarks. These evaluations are used as a "Go / No Go" determinant of whether teams will continue to receive funding. Thus, each team have a strong incentive to get as good a result as possible under the metric.

Each of the three GALE teams encompasses multiple sites and each has a collection of machine translation systems. A general strategy employed by all teams is to perform system combination over these systems to produce a synthetic translation that is better than the sum of its parts (Matusov et al., 2006; Rosti et al., 2007). The contribution of each component system is weighted by the expectation that it will produce good output. To our knowledge, none of the teams perform their own HTER evaluations in order to set these weights.

We evaluated the feasibility of using Mechanical Turk to perform HTER. We simplified the official GALE post-editing guidelines (NIST and LDC, 2007). We provided these instructions:

Edit Machine Translation *Your task is to edit the machine translation making as few changes as possible so that it matches the meaning of the human translation and is good English. Please follow these guidelines:*

- *Change the machine translation so that it has the same meaning as the human translation.*
- *Make the machine translation into intelligible English.*
- **Use as few edits as possible.**
- *Do not insert or delete punctuation simply to follow traditional rules about what is "proper."*
- *Please* **do not** *copy-and-paste the human translation into the machine translation.*

	Number of editors					
System	0	1	2	3	4	5
google.fr-en	.44	.29	.24	.22	.20	.19
google.de-en	.48	.34	.30	.28	.25	.24
rbmt5.de-en	.53	.41	.33	.28	.27	.25
geneva.de-en	.65	.56	.50	.48	.45	.45
tromble.de-en	.77	.75	.74	.73	.71	.70

Table 2: HTER scores for five MT systems. The edit rate decreases as the number of editors increases from zero (where HTER is simply the TER score between the MT output and the reference translation) and five.

We displayed 10 sentences from a news article. In one column was the reference English translation, in the other column were text boxes containing the MT output to be edited. To minimize the edit rate, we collected edits from five different Turkers for every machine translated segment. We verified these with a second HIT were we prompted Turkers to:

Judge edited translations *First, read the reference human translation. After that judge the edited machine translation using two criteria:*

- *Does the edited translation have the same meaning as the reference human translation?*
- *Is it acceptable English? Some small errors are OK, so long as its still understandable.*

For the final score, we choose the edited segment which passed the criteria and which minimized the edit distance to the unedited machine translation output. If none of the five edits was deemed to be acceptable, then we used the edit distance between the MT and the reference.

Setup We evaluated five machine translation systems using HTER. These systems were selected from WMT09 (Callison-Burch et al., 2009). We wanted a spread in quality, so we took the top two and bottom two systems from the German-English task, and the top system from the French-English task (which significantly outperformed everything else). Based on the results of the WMT09 evaluation we would expect the see the following ranking from the least edits to the most edits: google.fr-en, google.de-en, rbmt5.de-en, geneva.de-en and tromble.de-en.

Results Table 2 gives the HTER scores for the five systems. Their ranking is as predicted, indicating that the editing is working as expected. The table reports averaged scores when the five annotators are subsampled. This gives a sense of how much each additional editor is able to minimize the score for each system. The difference between the TER score with zero editors, and the HTER five editors is greatest for the rmbt5 system, which has a delta of .29 and is smallest for jhu-tromble with .07.

5.3 Reading comprehension

One interesting technique for evaluating machine translation quality is through reading comprehension questions about automatically translated text. The quality of machine translation systems can be quantified based on how many questions are answered correctly.

Jones et al. (2005) evaluated translation quality using a reading comprehension test the Defense Language Proficiency Test (DLPT), which is administered to military translators. The DLPT contains a collection of foreign articles of varying levels of difficulties, and a set of short answer questions. Jones et al used the Arabic DLPT to do a study of machine translation quality, by automatically translating the Arabic documents into English and seeing how many human subjects could successfully pass the exam.

The advantage of this type of evaluation is that the results have a natural interpretation. They indicate how understandable the output of a machine translation system is better than Bleu does, and better than other manual evaluation like the relative ranking. Despite this advantage, evaluating MT through reading comprehension hasn't caught on, due to the difficulty of administering it and due to the fact that the DLPT or similar tests are not publicly available.

We conducted a reading comprehension evaluation using Mechanical Turk. Instead of simply administering the test on Mechanical Turk, we used it for all aspects from test creation to answer grading. Our procedure was as follows:

Test creation We posted human translations of foreign news articles, and ask Tukers to write three questions and provide sample answers. We gave simple instructions on what qualifies as a good reading comprehension question.

Reading comprehension test *Please read the short newspaper article, and then write three reading comprehension questions about it, giving sample answers for each of your questions. Good reading comprehension questions:*

- *Ask about why something happened or why someone did something.*
- *Ask about relationships between people or things.*
- *Should be answerable in a few words.*

Poor reading comprehension questions:

- *Ask about numbers or dates.*
- *Only require a yes/no answer.*

Question selection We posted the questions for each article back to Mechanical Turk, and asked other Turkers to vote on whether each question was a good and to indicate if it was redundant with any other questions in the set. We sorted questions to maximize the votes and minimized redundancies using a simple perl script, which discarded questions below a threshold, and eliminated all redundancies.

Taking the test We posted machine translated versions of the foreign articles along with the questions, and had Turkers answer them. We ensured that no one would see multiple translations of the same article.

Answer questions about a machine translated text *You will answer questions about an article that has been automatically translated from another language into English. The translation contains many errors, but the goal is to see how understandable it is. Please do your best to guess at the right answers to the questions. Please:*

- *Read through the automatically translated article.*
- *Answer the questions listed below, using just a few words.*
- *Give your best guess at the answers, even if the translation is hard to understand.*
- *Don't use any other information to answer the questions.*

Grading the answers We aggregated the answers and used Mechanical Turk to grade them. We showed the human translation of the article, one question, the sample answer, and displayed all answers to it. After the Turkers graded the answers, we calculated the percentage of questions that were answered correctly for each system.

Turkers created 90 questions for 10 articles, which were subsequently filtered down to 47 good questions, ranging from 3–6 questions per article. 25 Turkers answered questions about each translated article. To avoid them answering the questions multiple times, we randomly selected which system's translation was shown to them. Each system's translation was displayed an average of 5 times per article. As a control, we had three Turkers answer the reading comprehension questions using the reference translation.

System	% Correct Answers
reference	0.94
google.fr-en	0.85
google.de-en	0.80
rbmt5.de-en	0.77
geneva.de-en	0.63
jhu-tromble.de-en	0.50

Table 3: The results of evaluating the MT output using a reading comprehension test

Table 3 gives the percent of questions that were correctly answered using each of the different systems' outputs and using the reference translation. The ranking is exactly what we would expect, based on the HTER scores and on the human evaluation of the systems in WMT09. This again helps to validate that the reading comprehension methodology. The scores are more interpretable than Blue scores and than the WMT09 relative rankings, since it gives an indication of how understandable the MT output is.

Appendix A shows some sample questions and answers for an article.

6 Conclusions

Mechanical Turk is an inexpensive way of gathering human judgments and annotations for a wide variety of tasks. In this paper we demonstrate that it is feasible to perform manual evaluations of machine translation quality using the web service. The low cost of the non-expert labor found on Mechanical Turk is cheap enough to collect redundant annotations, which can be utilized to ensure translation quality. By combining the judgments of many non-experts we are able to achieve the equivalent quality of experts.

The suggests that manual evaluation of translation quality could be straightforwardly done to validate performance improvements reported in conference papers, or even for mundane tasks like tracking incremental system updates. This challenges the conventional wisdom which has long held that automatic metrics must be used since manual evaluation is too costly and time-consuming.

We have shown that Mechanical Turk can be used creatively to produce quite interesting things.

We showed how a reading comprehension test could be created, administered, and graded, with only very minimal intervention.

We believe that it is feasible to use Mechanical Turk for a wide variety of other machine translated tasks like creating word alignments for sentence pairs, verifying the accuracy of document- and sentence-alignments, performing non-simulated active learning experiments for statistical machine translation, even collecting training data for low resource languages like Urdu.

The cost of using Mechanical Turk is low enough that we might consider attempting quixotic things like human-in-the-loop minimum error rate training (Zaidan and Callison-Burch, 2009), or doubling the amount of training data available for Urdu.

Acknowledgments

This research was supported by the EuroMatrix-Plus project funded by the European Commission, and by the US National Science Foundation under grant IIS-0713448. The views and findings are the author's alone.

A Example reading comprehension questions

Actress Heather Locklear arrested for driving under the influence of drugs

The actress Heather Locklear, Amanda on the popular series Melrose Place, was arrested this weekend in Santa Barbara (California) after driving under the influence of drugs. A witness saw her performing inappropriate maneuvers while trying to take her car out of a parking space in Montecito, as revealed to People magazine by a spokesman for the Californian Highway Police. The witness stated that around 4.30pm Ms. Locklear "hit the accelerator very roughly, making excessive noise and trying to take the car out from the parking space with abrupt back and forth maneuvers. While reversing, she passed several times in front of his sunglasses." Shortly after, the witness, who at first, apparently had not recognized the actress, saw Ms. Locklear stopping in a nearby street and leaving the vehicle.

It was this person who alerted the emergency services, because "he was concerned about Ms. Locklear's life." When the patrol arrived, the police found the actress sitting inside her car, which was partially blocking the road. "She seemed confused," so the policemen took her to a specialized centre for drugs and alcohol and submitted her a test. According to a spokesman for the police, the actress was cooperative and excessive alcohol was ruled out from the beginning, even if "as the officers initially observed, we believe Ms. Locklear was under the influences drugs." Ms. Locklear was arrested under suspicion of driving under the influence of some - unspecified substance, and imprisoned in the local jail at 7.00pm, to be released some hours later. Two months ago, Ms. Locklear was released from a specialist clinic in Arizona where she was treated after an episode of anxiety and depression.

4 questions were selected

- Why did the bystander call emergency services?
  He was concerned for Ms. Locklear's life.
- Why was Heather Locklear arrested in Santa Barbara?
  Because she was driving under the influence of drugs
- Where did the witness see her acting abnormally?
  Pulling out of parking in Montecito
- Where was Ms. Locklear two months ago?
  She was at a specialist clinic in Arizona.

5 questions were excluded as being redundant

- What was Heather Locklear arrested for?
  Driving under the influence of drugs
- Where was she taken for testing?
  A specialized centre for drugs and alcohol
- Why was Heather Locklear arrested?
  She was arested on suspicion of driving under the influence of drugs.
- Why did the policemen lead her to a specialized centre for drugs and alcohol
  Because she seemed confused.
- For what was she cured for two months ago?
  She was cured for anxiety and depression.

Answers to *Where was Ms. Locklear two months ago?* that were judged to be correct:

Arizona hospital for treatment of depression; at a treatmend clinic in Arizona; in the Arizona clinic being treated for nervous breakdown; a clinic in Arizona; Arizona, under treatment for depression; She was a patient in a clinic in Arizona undergoing treatment for anxiety and depression; In an Arizona mental health facility ; A clinic in Arizona.; In a clinic being treated for anxiety and depression.; at an Arizona clinic

These answers were judged to be incorrect: *Locklear was retired in Arizona; Arizona; Arizona; in Arizona; Ms.Locklaer were laid off after a treatment out of the clinic in Arizona.*

References

Bogdan Babych and Anthony Hartley. 2004. Extending the Bleu MT evaluation method with frequency weightings. In *Proceedings of ACL*.

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

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation (WMT08)*.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation (WMT09)*, March.

David Chiang, Steve DeNeefe, Yee Seng Chan, and Hwee Tou Ng. 2008. Decomposability of translation metrics for improved evaluation and efficient algorithms. In *Proceedings of EMNLP*.

Douglas Jones, Wade Shen, Neil Granoien, Martha Herzog, and Clifford Weinstein. 2005. Measuring translation quality by testing English speakers with a new defense language proficiency test for Arabic. In *Proceedings of the 2005 International Conference on Intelligence Analysis*.

LDC. 2005. Linguistic data annotation specification: Assessment of fluency and adequacy in translations. Revision 1.5.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing consensus translation for multiple machine translation systems using enhanced hypothesis alignment. In *Proceedings of EACL*.

NIST and LDC. 2007. Post editing guidelines for GALE machine translation evaluation. Guidelines developed by the National Institute of Standards and Technology (NIST), and the Linguistic Data Consortium (LDC).

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

Michael Paul. 2006. Overview of the IWSLT 2006 evaluation campaign. In *Proceedings of International Workshop on Spoken Language Translation*.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. 2007. Combining outputs from multiple machine translation systems. In *Proceedings of HLT/NAACL*.

Markus Schulze. 2003. A new monotonic and clone-independent single-winner election method. *Voting Matters*, (17), October.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast - but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*.

Omar F. Zaidan and Chris Callison-Burch. 2009. Feasibility of human-in-the-loop minimum error rate training. In *Proceedings of EMNLP*.

How well does active learning *actually* work? Time-based evaluation of cost-reduction strategies for language documentation.

Jason Baldridge
Department of Linguistics
The University of Texas at Austin
jbaldrid@mail.utexas.edu

Alexis Palmer
Computational Linguistics
Saarland University
apalmer@coli.uni-sb.de

Abstract

Machine involvement has the potential to speed up language documentation. We assess this potential with timed annotation experiments that consider annotator expertise, example selection methods, and suggestions from a machine classifier. We find that better example selection and label suggestions improve efficiency, but effectiveness depends strongly on annotator expertise. Our expert performed best with uncertainty selection, but gained little from suggestions. Our non-expert performed best with random selection and suggestions. The results underscore the importance both of measuring annotation cost reductions with respect to time and of the need for cost-sensitive learning methods that adapt to annotators.

1 Introduction

Data annotated with linguistically interesting labels is used in a wide variety of contexts. Computational linguists generally use annotated data as training and evaluation material for natural language processing systems; corpus linguists use it to test hypotheses about language; documentary linguists create interlinear glossed texts to preserve examples of endangered languages and hypotheses about the grammars of those languages. Regardless of the context, creating annotated data is costly in terms of time and/or money. Since both time and money are undeniably in limited supply, there is a widely shared desire to reduce this cost.

Reducing cost involves strategies that do more with fewer human-annotated labels and/or reduce the per-label cost. An example of the former is active learning, which focuses annotation effort on data points selected by the learner(s) for their expected utility in developing a more accurate model (Settles, 2009). Examples of the latter include providing suggestions from a machine labeler and using extremely cheap human labelers, e.g. with the Amazon Mechanical Turk (Snow et al., 2008). Different techniques may be more or less applicable depending on the language being annotated, the kind of labels which are desired (tags, syntactic structures, etc.), and the desired use of the annotated data (e.g., for training models, testing linguistic hypotheses, or preserving a language).

This paper discusses experiments that measure the effectiveness of machine-aided annotation for language documentation using both active learning simulation experiments and annotation experiments which involve actual documentary linguists interacting with machine example selection and label suggestion. Specifically, we deal with the task of labeling morphemes of the Mayan language Uspanteko with fine-grained parts-of-speech. We also run active learning simulation experiments for part-of-speech tagging for Danish, Dutch, English, Swedish, and Uspanteko to show the validity of our models and methods in a standard setting. For Uspanteko, we provide results from annotation experiments in which annotation cost is measured in terms of the actual annotation time required while varying three factors: (1) example selection, (2) machine label suggestions, and (3) annotator expertise.

Our findings indicate that there is considerable promise for reducing the cost of producing IGT, but they also demonstrate considerable variation due to the interaction of these factors. This suggests different prescriptions for appropriate strategies in different contexts. Most clearly, the worst performing strategy—by far—is that used in nearly all documentary work: sequential annotation without automation. Also, our expert annotator did best with examples picked by uncertainty selection, while our non-expert did best with random selection aided by machine label sug-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 296–305,
Singapore, 6-7 August 2009.

Language	#words-tr	#words-dev	#tags	#sents-tr	#sents-dev	Avg.sent	Avg.tr.sent	Avg.dev.sent
Danish	62825	31561	10	3570	1618	18.18	17.60	19.50
Dutch	129586	65483	13	9365	3982	14.61	13.84	16.44
English	167593	131768	45	6945	5527	24.00	24.13	23.84
Swedish	127684	63783	41	7326	3714	17.34	17.43	17.17
Uspanteko	43473	19906	69	7423	3288	5.92	5.86	6.05

Table 1: Corpora: number of words and sentences, number of possible tags, and average sentence length.

gestions. This difference confirms the importance of cost-sensitive active learning strategies that are not just learner-guided, but also take into account modeling of the annotators (Settles et al., 2008; Haertel et al., 2008; Vijayanarasimhan and Grauman, 2008). Finally, we confirm the importance of using actual annotation time to measure annotation cost: a unit-cost assumption—even at a fine-grained level—can dramatically misrepresent the actual effectiveness of different strategies.

2 Task and data

Annotation task: language documentation The amount of money spent on obtaining human annotations is an extremely important concern in much language annotation. However, there is a further urgency for annotation in the case of language documentation: languages are dying at the rate of two each month. By the end of this century, half of the approximately 6000 extant spoken languages will cease to be transmitted effectively from one generation of speakers to the next (Crystal, 2000). Recorded and transcribed texts annotated with detailed linguistic information create an important multi-faceted record of these languages, but there are few trained linguists with adequate time and appropriate levels of funding relative to the size of the problem. Annotation cost—in both time and money—is thus keenly felt in the work of documenting and describing endangered languages. Active learning and automated label suggestions could help deal with this language documentation bottleneck.

We focus on one stage of language documentation, the production of interlinear glossed text (IGT), a standard form of annotation that involves both morphological and grammatical analysis. IGT is generally created following transcription and translation of recorded speech, with the annotations often being provided by trained annotators with varying levels of expertise. The result is generally a small amount of IGT annotated data and a greater amount of unannotated data.

Data We use a collection of 32 interlinear glossed texts (IGT) in the Mayan language Uspanteko. This corpus was cleaned up and adapted by Palmer et al. (2009) from an original collection of 67 texts that were collected, transcribed, translated and annotated by the OKMA language documentation project (Pixabaj et al., 2007).

Two core tasks in creating IGT are morphological analysis and tagging morphemes with their glosses (labels indicating part-of-speech and/or grammatical function). We deal with the latter task and assume texts are morphologically segmented. Standard four-line IGT has morphemes on one line and their glosses on the next. The gloss line includes labels for grammatical morphemes (e.g. `PL` or `COM`) and translations of stems (e.g. *hablar* or *idioma*). The following is an Uspanteko example:

(1)
TEXT: `Kita' tinch'ab'ej laj inyolj iin`
MORPH: `kita' t-in-ch'abe-j laj in-yol-j iin`
GLOSS: NEG INC-E1S-hablar-SC PREP A1S-idioma-SC yo
POS: PART TAM-PERS-VT-SUF PREP PERS-S-SUF PRON
TRANS: 'No le hablo en mi idioma.'

We use a single layer that is a combination of the GLOSS and POS layers (Palmer et al., 2009). For (1), the morphemes and labels for our task are:

(2)
kita' t- in- ch'abe -j laj in- yol -j iin
NEG INC E1S VT SC PREP A1S S SC PRON

We also consider POS-tagging for Danish, Dutch, English, and Swedish; the English is from sections 00-05 (as training set) and 19-21 (as development set) of the Penn Treebank (Marcus et al., 1993), and the other languages are from the CoNLL-X dependency parsing shared task (Buchholz and Marsi, 2006).[1] We split the original training data into training and development sets. Table 1 shows the number of words and sentences in each split of each dataset, as well as the number of possible labels and the average sentence length. The Uspanteko data is counted in morphemes rather than words; also, the Uspanteko texts are divided at the clause rather than sentence level. This gives the corpus a much lower average clause length than the other languages (Table 1).

[1]The subset of the Penn Treebank was chosen to be of comparable size to the CoNLL datasets.

3 Model and methods

Classification model. We use a standard maximum entropy classifier for tagging Danish, Dutch, English, and Swedish words with POS-tags and tagging Uspanteko morphemes with Gloss/POS tags. The label for a word/morpheme is predicted based on the word/morpheme itself plus a window of two units before and after. Standard part-of-speech tagging features (Ratnaparkhi, 1998; Curran and Clark, 2003) are extracted from the morpheme to help with predicting labels for previously unseen morphemes. This is a strong but standard model; better, more complex models could be used, but the gains are likely to be small. Thus, we opted for simplicity in our model so as to focus more on the interaction between the annotator and different levels of machine involvement.

The accuracy of the tagger on the datasets when trained on all available training material is given in the following table, along with accuracy of a unigram model (learned from the training set and constrained by a tag dictionary for known words).

	Unigram	Model
Danish	91.62%	95.58%
Dutch	90.92%	93.57%
English	87.87%	93.25%
Swedish	84.91%	87.74%
Uspanteko	77.84%	79.39%

Sample selection. We consider three sample selection methods: **seq**uential, **rand**om, and **unc**ertainty. Sequential selection is important to consider as it is the default in documentary projects. It is sub-optimal for corpora with contiguous sub-domains, since it necessitates working through many similar examples before getting to possibly more informative examples. Random selection is a model-free method that avoids the sub-domain trap by sampling freely from the entire corpus. It generally works better than sequential selection and provides a strong baseline against which to compare learner-guided selection.

Uncertainty selection (Cohn et al., 1995) identifies examples the model is least confident about. We measure uncertainty as the entropy of the label distribution predicted by the maximum entropy model for each example. Uncertainty for a clause is calculated as the average entropy per morpheme; clauses with the highest average entropy are selected for labeling.

A recent development in active learning is cost-sensitive selection that is guided not only by the learner but also by the expected cost of labeling an example based on its likely complexity and/or the reliability of the annotator. Settles et al. (2008) provide empirical validation for cost-related intuitions; for example, that cost of annotation is static neither per example nor per annotator. Also, they show that taking annotation cost into account can improve active learning effectiveness, but that learning to predict annotation cost is not yet well-understood. A cost-sensitive Return on Investment heuristic is developed in Haertel et al. (2008) and tested in a simulated POS-tagging context. Our experiments do not employ cost-sensitive selection, but our results—from live (non-simulated) active learning experiments of real-world scale—empirically support the need to consider cost-sensitive selection if better cost reductions are to be achieved.

Annotation setup. We compare results from two annotators with different levels of exposure to Uspanteko. Both are documentary linguists with extensive field experience. Our **expert annotator** is a native speaker of K'ichee', a closely related Mayan language, and has worked extensively on Uspanteko. Our **non-expert annotator** had no prior experience with Uspanteko and only limited exposure to Mayan languages. During annotation, he used an Uspanteko-Spanish dictionary.

For each selection method, we consider two conditions for providing classifier labels: a **do-suggest** (**ds**) condition where the labels predicted by the machine learner are shown to the annotator, and a **no-suggest** (**ns**) condition where the annotator does not see the predictions. With **ds**, the annotator is shown the most probable label and a ranked list of all labels assigned a probability greater than half that of the best label. For **ns**, the annotator sees a frequency-ranked list of labels previously seen in training data for the given morpheme.

Annotators improve as they see more examples. To minimize the impact of this learning process, annotation is done in rounds. Each round consists of sixty clauses—six batches of ten each for the six experimental cases. The annotator is free to break between batches. Following annotation, the newly-labeled clauses are added to the training data, and a new model is trained and evaluated. Both annotators completed fifty-six rounds of annotation. See Palmer et al. (2009) for more details on the annotation setup.

Measuring annotation cost. Active learning studies usually *simulate* annotation and use a unit cost assumption that each word, sentence, constituent, document, etc. takes the same time to annotate. This is often the only option since corpora typically do not retain annotation time, but it is likely to exaggerate the annotation cost reductions achieved. This is exacerbated with active learning: the informative examples it seeks to find are typically harder to annotate (Hachey et al., 2005).

Baldridge and Osborne (2008) correlate a unit cost in terms of *discriminants* (decisions made by annotators about valid parses) to annotation time. This is a better approximation than unit costs where such a relationship cannot be established. However, it is based on a *static* measurement of annotation time, and clearly the time taken to annotate an example is not a function of the example alone. Annotation time is actually *dynamic* in that it is dependent on how many and what kinds of examples have already been annotated. An "informative" example is likely to take longer to annotate if selected early than it would after the annotator has seen many other examples.

Thus, it is important to measure annotation time *embedded in the context* of a particular annotation experiment with the sample selection/labeling strategies of interest. In our annotation experiments, we measure the exact time taken to annotate each example by each annotator and use this as the cost metric, inspired by Ngai and Yarowsky (2000). In the simulation studies, as we are unable to measure time, we measure cost by sentence/clause and word/morpheme.

Learning curve comparison. We are interested in comparative evaluation of many different experimental settings, across which we vary selection methods, use of label suggestions, and annotators. To achieve this, it is useful to have a summary value for comparing the results from two individual experiments. One such measure is the percentage error reduction (PER), measured over a discrete set of points on the first 20% of the points on the learning curve (Melville and Mooney, 2004).[2]

We use a new related measure, which we call the *overall* percentage error reduction (**OPER**), that uses the *entire* area under the curves given by fitted nonlinear regression models rather than averaging over a subset of data points. Specifically, we fit a modified Michaelis-Menton model:

$$f(cost, (K, V_m, A)) = \frac{V_m(A + cost)}{K + cost}$$

The (original) parameters V_m and K respectively correspond to the horizontal asymptote and the cost where accuracy is halfway between 0 and V_m. The additional parameter A allows for a better fit to our data by allowing for less sharp elbows and letting $cost$ be zero. Model parameters were determined with `nls` in R (Ritz and Streibig, 2008).

With the fitted regression models, it is straightforward to calculate the area under the curve between a start cost c_i and end cost c_j by taking the integral from c_i to c_j. The overall accuracy for the experiment is given by dividing that area by $100 \times (c_j - c_i)$. Call this the overall curve accuracy (OCA). Then, for experiment A compared to experiment B, OPER(A,B) = $\frac{OCA_A - OCA_B}{100 - OCA_B}$. For the simulation experiments we calculate OPER for only the first 20% of cost units, like Melville and Mooney. For the annotation experiments, we calculate it for the minimum amount of time spent on any of the experiments (which ended up using less than 10% of all available morphemes).

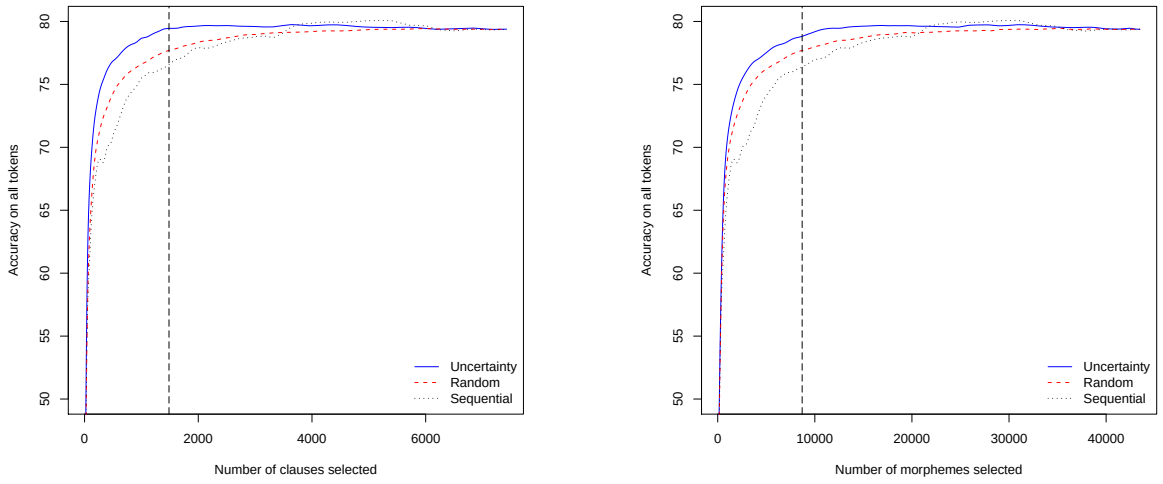
4 Simulation experiments

We verify that our tagger and dataset behave as expected in standard active learning experiments by running simulations on the Uspanteko data set, and on POS-tagging for Danish, Dutch, English, and Swedish. Here, we vary only the selection method: **seq**uential, **rand**om, or **unc**ertainty.

For each language, we randomly select a seed set of 10 labeled sentences. The number of examples selected to be labeled in each round begins at 10 and doubles after every 20 rounds. For **rand** and **unc**, each batch of examples is selected from a pool (size of 1000) that is itself randomly selected from the entire set of remaining unlabeled examples. **rand** and **unc** experiments for each language are replicated 5 times; splines and regressions are computed over all runs for each condition.

Figure 1 gives learning curves for the Uspanteko simulations, with cost measured in terms of (a) clauses and (b) morphemes. Both graphs show the usual behavior found in active learning experiments. **rand** and **unc** both rise more quickly than **seq**, and **unc** is well above **rand**. The relationship between the methods is the same regardless

[2]This is justified in standard conditions, sampling from a finite corpus: active learning runs out of interesting examples after considering a fraction of the data, so the curve is *artificially* pulled down by the remaining, boring examples.



(a) (b)

Figure 1: Learning curves for simulations; (a) clause cost and (b) morphemes cost. The dashed vertical lines indicate (a) #clauses=1485 and (b) #morphemes=8695 (to compare OPER values).

	$\frac{\textbf{rand}}{\textbf{seq}}$	$\frac{\textbf{unc}}{\textbf{seq}}$	$\frac{\textbf{unc}}{\textbf{rand}}$
Uspanteko-Clauses	5.86	13.27	7.86
Uspanteko-Morphs	7.47	11.68	4.55

Table 2: OPER values for Uspanteko simulations, comparing **clause** and **morpheme** cost. $\frac{A}{B}$ indicates we compute OPER(A,B).

	$\frac{\textbf{rand}}{\textbf{seq}}$	$\frac{\textbf{unc}}{\textbf{seq}}$	$\frac{\textbf{unc}}{\textbf{rand}}$
Danish	4.58	6.95	2.48
Dutch	21.95	23.68	2.20
English	6.55	8.00	1.56
Swedish	9.56	9.29	-0.30
Uspanteko	7.47	11.68	4.55

Table 3: OPER values for **morpheme** cost for simulations. $\frac{A}{B}$ indicates we compute OPER(A,B).

of the cost metric, but the relative differences in cost-savings are not, which we see when we look at OPER values.

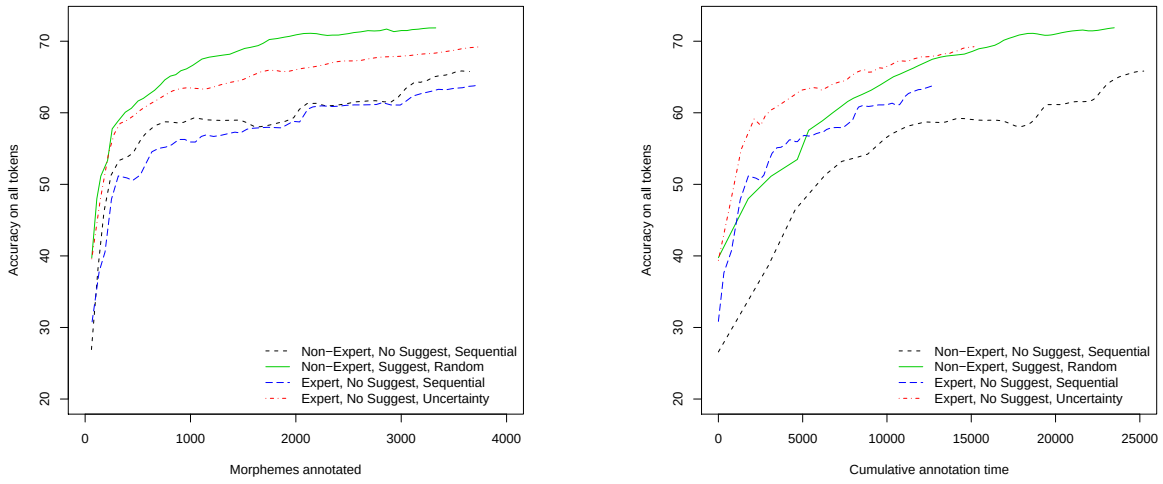
The dashed vertical lines in the two graphs correspond to the 20% mark used to calculate OPER values, which are given in Table 2. Most importantly, note the much larger OPER for **unc** over **rand** with clause cost (7.86 vs 4.55). Also note that OPER(**rand**,**seq**) is *lower* with clause cost—this indicates that the beginning portions of the corpus contain longer sentences with more morphemes, an accident which overstates how well **seq** would likely work in general.

Since **rand** is unbiased with respect to picking longer sentences, the large increase of OPER(**unc**,**rand**) from 4.55 to 7.86 is a clear indication of the well-known—but not always attended to—tendency of uncertainty sampling to select longer sentences. Consequently, one should at least use sub-sentence cost in order not to overstate the gains from active learning. The annotation experiments in the next section take this word of caution one step further: even sub-sentence cost (morpheme cost, in our setting) can overestimate gains since the morphemes selected are actually harder to annotate and thus take more time.

Table 3 gives overall percentage error reductions (OPER) between different selection methods based on word/morpheme cost, for each language. For all languages, **rand** and **unc** are better than **seq**. Only in the case of Swedish is there no benefit from **unc** over **rand**. For Dutch, the large gains over **seq** for both **rand** and **unc** accurately reflect the heterogeneity of the underlying Alpino corpus.[3] Most importantly, for Uspanteko, there are large reductions from **unc** to **rand** to **seq**, mirroring the clear trends in Figure 1b.

These simulations have an unrealistic "perfect" annotator, the corpus. Next, we discuss results with real annotators—who may be fallible or may (reasonably) beg to differ with the corpus analysis.

[3] `http://www.let.rug.nl/vannoord/trees/`



(a) (b)

Figure 2: A sample of the learning curves with (a) morpheme cost and (b) time cost. Morpheme cost ranks strategies for a given annotator similarly to time cost, but it gives dramatically different results from time cost when used to compare different annotators.

5 Annotation experiments

With two annotators (**expert**, **non-expert**), three selection methods (**seq**, **rand**, **unc**), and two machine labeling settings (**ns**, **ds**), we obtain 12 different experiments. Each experiment measures accuracy in terms of all words and unknown words and cost in terms of clauses, morphemes and time; this produces six views on every experiment. In this paper we focus on one view: accuracy over all words with time-based evaluation of cost.

As with the simulations, clause cost in the annotation experiments overestimates the cost reductions. For morpheme cost, the annotation experiments show that (a) it also overstates cost reductions compared to time, and (b) it can mis-state relative effectiveness when comparing annotators.

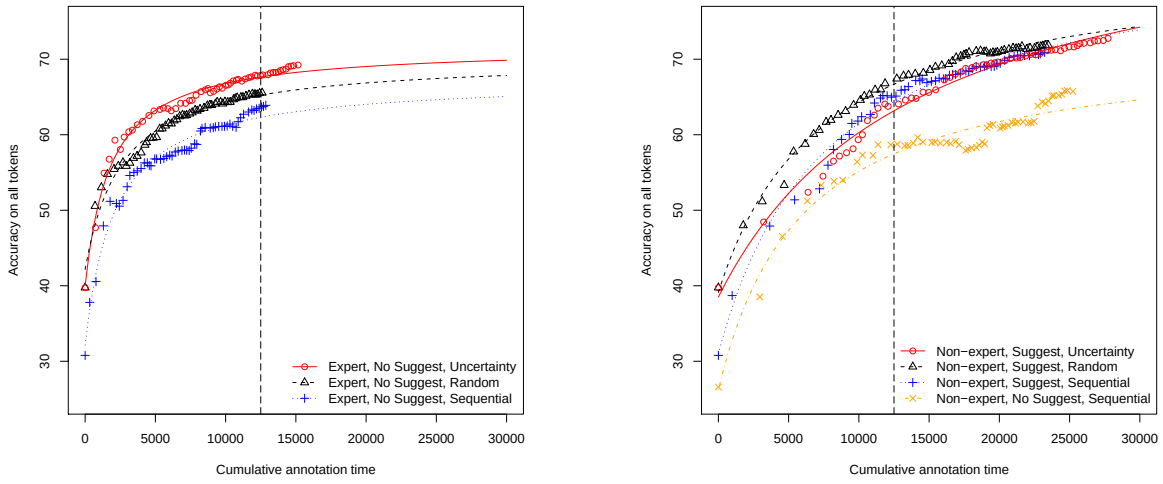
The big picture. Figure 2 shows curves for four experiments: **seq**-**ns** for both annotators[4] and the most effective overall condition for each annotator. Figure 2a uses morpheme cost evaluation; on that metric, both annotators appear to be about equally effective with **seq**-**ns** and much more effective with machine involvement (**unc** or **ds**) than without. Additionally, the non-expert's **rand**-**ds** appears to beat the expert's **unc**-**ns**. However, the time cost evaluation in Figure 2b tells a dramatically different story. Each annotator's machine-involved experiment is much better than their **seq**-**ns**, but now the expert's best is clearly better than the non-expert's. We see this as clear evidence for the need for cost-sensitive learning over vanilla active learning (as we do here).[5]

The non-expert with **rand**-**ds** caught up to and surpassed the unaided expert in about six hours total annotation time, and he caught up to her **unc**-**ns** curve after 35 hours. This is encouraging since often language documentation projects have participants with a wide range of expertise levels, and these results suggest that assistance from machine learning, if done properly, may increase the effectiveness of participants with less language-specific expertise. We are also encouraged, with respect to the effectiveness of active learning, that the expert's best performance is obtained with uncertainty-based selection.

Within annotator comparisons. Figure 3 shows both actual measurements and the fitted nonlinear regression curves used to compute OPER. Figure 3a, the expert without suggestions, exhibits typical active learning behavior similar to that seen in the simulation experiments. Figure 3b,

[4]Recall that sequential annotation is the default mode for producing IGT, so this strategy is of particular interest.

[5]It is also clear to see that, unsurprisingly, the expert spent much less time to complete the 56 rounds than the non-expert. In general, the expert annotator was much quicker, particularly in early rounds, averaging 4.1 seconds per morpheme annotated against the non-expert's 8.0 second average. See Palmer et al. (2009) for more details.



(a) (b)

Figure 3: Sample measurements and fitted nonlinear regression curves for (a) the expert and (b) the non-expert. Note that the scale is consistent for comparability. The dashed vertical lines indicate 12,500 seconds (about 35 hours), which is the upper limit used in computing OPER values for Table 4.

the non-expert *with* suggestions, shows that in the **ds** conditions the non-expert was less effective with **unc**. This is not unexpected: uncertainty selects harder examples that will either take longer to annotate or are easier to get wrong, especially if the annotator trusts the classifier and *especially* on examples the classifier is uncertain about. Nonetheless, in all **ds** cases, the non-expert performs better than with **seq-ns**.

OPER. Table 4 provides OPER values from time 0 to 12,500 seconds (about 35 hours), the minimum amount of annotation time logged in any one of the twelve experiments.[6] The table mixes three types of comparison: (1) the boxed values on the diagonal give OPER for the expert versus the non-expert given the same selection and suggestion conditions; (2) the upper (right) triangle gives OPER for the expert versus herself for different conditions; and (3) the lower (left) triangle is the non-expert versus himself. For example: (1) the expert obtained an 11.52 OPER versus the non-expert when both used **rand-ns**; (2) the expert obtained a 10.52 OPER by using **rand-ds** rather than **seq-ns**; and (3) the non-expert obtained a 5.93 OPER over **rand-ns** by using **rand-ds**.

A number of patterns emerge. Quite unsurprisingly, the values on the diagonal show that the expert is more effective than the non-expert in all conditions. Also, every other condition is more effective than **seq-ns** for both annotators (first row for the expert, first column for the non-expert). **unc-ns** and **rand-ds** are particularly effective for the non-expert, giving OPERs of 19.20 and 18.59 over **seq-ns**, respectively. These reductions, bigger than the expert's reductions of 14.17 and 10.52 for the same conditions, considerably reduce the large gap in **seq-ns** effectiveness between the two annotators (see Figure 2b).

The expert actually gains very little from **ds** for both **rand** and **unc**: adding suggestions gave OPERs of just 1.83 and .39, respectively. In contrast, the non-expert obtains an improvement of 5.93 OPER when suggestions are used with **rand**,

non-exp \ exp	seq-ns	rand-ns	unc-ns	seq-ds	rand-ds	unc-ds
seq-ns	15.99	8.85	14.17	6.34	10.52	14.50
rand-ns	13.46	11.52	5.83	-2.76	1.83	6.20
unc-ns	19.20	6.63	10.76	-9.12	-4.25	0.39
seq-ds	10.24	-3.72	-11.09	12.34	4.46	8.72
rand-ds	18.59	5.93	-0.76	9.30	7.67	4.45
unc-ds	11.19	-2.62	-9.91	1.06	-9.09	19.13

Table 4: Overall percentage error reduction (OPER) comparisons, with timing cost. See explanation of table in the **OPER** subsection.

[6] Stopping at 12,500 seconds ensures a fair comparison, for example, between the expert and the non-expert because it requires no extrapolation of the expert's performance.

but performs *worse* when used with **unc** (-9.91 OPER). Even more striking: the non-expert's **unc-ds** is worse than **rand-ns** (-2.62 OPER), a completely model-free setting. These variations demonstrate the importance of modeling annotator fallibility and sensitivity to cost, as well as characteristics of the annotation task itself, if learner-guided selection and suggestion are to be used (Donmez and Carbonell, 2008; Arora et al., 2009).

Annotator accuracy. Another factor which must be considered when annotation is done by human annotators (rather than being simulated) is the accuracy of the humans' labels. Table 5 shows the overall accuracy of the annotators' labels for each condition (after 56 rounds) as measured against the original OKMA annotations. Unsurprisingly, **unc** selection picks examples that are more difficult to annotate: accuracy for both annotators suffers in both **unc-ns** and **unc-ds**.

It may seem surprising that the non-expert's accuracies are generally higher than the expert's. The main reason for this is that the non-expert took nearly twice as long to annotate his examples, so each one was done with more care. However, this difference also highlights challenges that arise when we bring active learning into non-simulated annotation contexts. The typical assumption is that gold standard labeled data represents a true, fixed target, against which annotator or machine-predicted labels should be measured. In language documentation, though, the analysis of the language is continually evolving, and analysis and annotation each inform the other. In fact, the expert recognized (in the morphological segmentation) several linguistic phenomena for which the analysis has changed since the original OKMA annotations were done. As she changed her analyses, her labels diverged from those of the original corpus—another reason for her "lower" accuracy. This is to say that the ground truth of the current OKMA annotations we had to work with can be viewed as one (valid) stage in the iterative reanalysis process that language documentation is.

Error analysis. Preliminary analysis of 'errors' made by the annotators supports the idea that the results seen in Table 5 are heavily influenced by changes in the expert's analysis of the language. Some duplicate clause annotation occurred for each annotator, because each of the twelve annotator-selection-suggestion conditions drew from the same global set of unlabeled examples. This duplication allows us to measure the consistency of each annotator on labeling such duplicate clauses. Table 6 shows the percentage of morphemes labeled consistently by each annotator. Numbers for the expert appear in the top (right) triangle, and for the non-expert in the bottom (left) triangle. Overall intra-annotator consistency is much higher for the expert (88.38%) than for the non-expert (81.64%), suggesting that the expert maintained a more consistent mental model of the language, but one which disagrees in some areas with the original annotations.

	expert	**non-expert**
seq-ns	73.17%	75.09%
rand-ns	69.90%	74.37%
unc-ns	61.23%	60.04%
seq-ds	67.48%	73.13%
rand-ds	68.34%	73.03%
unc-ds	59.79%	60.27%

Table 5: Overall accuracy of annotators' labels, measured against OKMA annotations.

Another key error source comes from differences in use of one individual label: the annotators could assign a label that does not appear in the original corpus. This is yet another issue that does not—in fact, *cannot*—arise in simulated active learning. The label **ESP** was introduced for labeling Spanish loans or insertions (such as the discourse marker *entonces*) which do not have a clear function in Uspanteko grammar. Such tokens are inconsistently labeled in the original corpus, usually with catch-all categories like particle or adverb. The annotators felt that the best analysis was to mark the tokens as of Spanish origin. The expert annotator used the **ESP** label for 2086 of 24129 tokens (8.65%) versus 221 of 22819 tokens (0.97%) for the non-expert. Any such token labeled with **ESP** is scored as incorrect when compared to the OKMA standard, so this label alone accounts for more than 7% of the expert annotator's total error.

Finally, Table 7 presents inter-annotator agreement measured as percent agreement on morphemes in clauses labeled by both annotators. Note that in general agreement seems to be lowest for clauses duplicated in **unc** conditions, supporting the expected result that uncertainty-based selection does indeed select clauses that are more difficult for human annotators to label.

non \ exp	seq-ns	rand-ns	unc-ns	seq-ds	rand-ds	unc-ds
seq-ns	—	95.00% (41)	87.10% (56)	92.39% (60)	91.02% (28)	88.83% (51)
rand-ns	90.11% (49)	—	90.91% (57)	87.57% (35)	90.94% (50)	89.53% (57)
unc-ns	80.80% (44)	81.68% (54)	—	81.35% (41)	89.10% (40)	87.82% (332)
seq-ds	90.00% (54)	87.94% (44)	77.97% (48)	—	86.13% (42)	82.14% (42)
rand-ds	90.15% (52)	86.64% (45)	79.46% (62)	81.43% (44)	—	87.06% (49)
unc-ds	84.15% (47)	78.55% (52)	77.68% (328)	78.81% (35)	77.95% (60)	—

Table 6: Annotation consistency, expert and non-expert, (number of duplicate clauses, of 560 possible)

non \ exp	seq-ns	rand-ns	unc-ns	seq-ds	rand-ds	unc-ds
seq-ns	69.91% (523)	70.82% (42)	62.42% (48)	72.35% (54)	74.25% (28)	67.82% (47)
rand-ns	71.32% (48)	83.94% (39)	66.56% (47)	66.15% (43)	73.75% (42)	67.55% (52)
unc-ns	66.31% (48)	67.87% (53)	62.31% (301)	58.87% (51)	73.31% (40)	61.10% (298)
seq-ds	73.35% (60)	75.56% (34)	56.39% (37)	60.02% (540)	66.00% (44)	61.01% (36)
rand-ds	68.67% (50)	76.40% (63)	66.67% (58)	65.88% (47)	76.33% (42)	66.99% (64)
unc-ds	65.41% (50)	67.98% (55)	60.43% (263)	58.13% (38)	70.74% (57)	60.40% (275)

Table 7: IAA: expert v. non-expert, percentage of morphemes in agreement, (number of duplicate clauses, of 560 possible)

6 Conclusion

Through actual annotation experiments that control for several factors, we have evaluated the potential of incorporating active learning and label suggestions to speed up morpheme glossing in a realistic language documentation context. Some configurations of learner-guided example selection and machine label suggestions perform far better than the standard strategy of sequential selection without suggestions. However, the effectiveness of any given strategy depends on annotator expertise. The impact of differences between annotators directly bears on the point made by Donmez and Carbonell (2008) that if cost reductions are to be reliably obtained with active learning techniques, annotators' fallibility, unreliability, and sensitivity to cost must be modeled.

Our results suggest some possible prescriptions for tuning techniques according to annotator expertise. However, even if we can estimate a relative level of expertise, following such broad prescriptions is unlikely to be more robust than an approach which *adapts* selection and suggestion to the individual annotator, perhaps working within an annotation group. Indeed, it seems that dealing with variation in annotators/oracles may be more important than devising better selection strategies.

The difference in performance due to expertise suggests that using multiple annotators to check relative annotation rate and accuracy of different annotators could be a key ingredient in any actually deployed active learning system. This could provide for better modeling of individual annotators as part of an annotation group they can be compared against, allowing the system, for example, to throttle active selection if an annotator appears to be too slow or inaccurate.

Another major issue we highlight is the uncertainty around the question of whether active learning works in practical applications. Respondents to the survey of Tomanek and Olsson (2009) indicated that this uncertainty—will active learning work? what methods or techniques will work best?—is one of the reasons active learning is not widely used in actual annotation. In addition, creating the necessary software infrastructure to build an active learning enabled annotation system—a system which must interface robustly between data, annotator, and machine classifier, yet still be easy to use—is a substantial hurdle. It seems unlikely that there will be much uptake until a) consistent, large cost reductions can be shown in actual annotation studies, and b) appropriate, *tunable*, widely-available software exists.

Acknowledgments

This work is funded by NSF grant BCS 06651988 "Reducing Annotation Effort in the Documentation of Languages using Machine Learning and Active Learning." Thanks to Eric Campbell, Katrin Erk, Michel Jacobson, Taesun Moon, Telma Kaan Pixabaj, and Elias Ponvert.

References

Shilpa Arora, Eric Nyberg, and Carolyn P. Rosé. 2009. Estimating annotation cost for active learning in a multi-annotator environment. In *Proceedings of the NAACL HLT Workshop on Active Learning for Natural Language Processing*, pages 18–26, Boulder, CO.

Jason Baldridge and Miles Osborne. 2008. Active learning and logarithmic opinion pools for HPSG parse selection. *Natural Language Engineering*, 14(2):199–222.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June. Association for Computational Linguistics.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1995. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press.

David Crystal. 2000. *Language Death*. Cambridge University Press, Cambridge.

James R. Curran and Stephen Clark. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 10th Conference of the European Association for Computational Linguistics*, pages 91–98.

Pinar Donmez and Jaime G. Carbonell. 2008. Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of CIKM08*, Napa Valley, CA.

Ben Hachey, Beatrice Alex, and Markus Becker. 2005. Investigating the effects of selective sampling on the annotation task. In *Proceedings of the 9th Conference on Computational Natural Language Learning*, Ann Arbor, MI.

Robbie A. Haertel, Kevin D. Seppi, Eric K. Ringger, and James L. Carroll. 2008. Return on investment for active learning. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational linguistics*, 19:313–330.

Prem Melville and Raymond J. Mooney. 2004. Diverse ensembles for active learning. In *Proceedings of the 21st International Conference on Machine Learning*, pages 584–591, Banff, Canada.

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

Alexis Palmer, Taesun Moon, and Jason Baldridge. 2009. Evaluating automation strategies in language documentation. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 36–44, Boulder, CO.

Telma Can Pixabaj, Miguel Angel Vicente Méndez, María Vicente Méndez, and Oswaldo Ajcot Damián. 2007. Text Collections in Four Mayan Languages. Archived in The Archive of the Indigenous Languages of Latin America.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Christian Ritz and Jens Carl Streibig. 2008. *Nonlinear Regression with R*. Springer.

Burr Settles, Mark Craven, and Lewis Friedland. 2008. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*.

Burr Settles. 2009. Active learning literature survey. Technical Report Computer Sciences Technical Report 1648, University of Wisconsin-Madison.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast - but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP 2008*, pages 254–263.

Katrin Tomanek and Fredrik Olsson. 2009. A Web Survey on the Use of Active learning to support annotation of text data. In *Proceedings of the NAACL HLT Workshop on Active Learning for Natural Language Processing*, pages 45–48, Boulder, CO.

Sudheendra Vijayanarasimhan and Kristen Grauman. 2008. Multi-level active prediction of useful image annotations for recognition. In *Proceedings of NIPS08*, Vancouver, Canada.

Automatically Evaluating Content Selection in Summarization without Human Models

Annie Louis
University of Pennsylvania
lannie@seas.upenn.edu

Ani Nenkova
University of Pennsylvania
nenkova@seas.upenn.edu

Abstract

We present a fully automatic method for content selection evaluation in summarization that does not require the creation of human model summaries. Our work capitalizes on the assumption that the distribution of words in the input and an informative summary of that input should be similar to each other. Results on a large scale evaluation from the Text Analysis Conference show that input-summary comparisons are very effective for the evaluation of content selection. Our automatic methods rank participating systems similarly to manual model-based pyramid evaluation and to manual human judgments of responsiveness. The best feature, Jensen-Shannon divergence, leads to a correlation as high as 0.88 with manual pyramid and 0.73 with responsiveness evaluations.

1 Introduction

The most commonly used evaluation method for summarization during system development and for reporting results in publications is the automatic evaluation metric ROUGE (Lin, 2004; Lin and Hovy, 2003). ROUGE compares system summaries against one or more model summaries by computing n-gram word overlaps between the two. The wide adoption of such automatic measures is understandable because they are convenient and greatly reduce the complexity of evaluations. ROUGE scores also correlate well with manual evaluations of content based on comparison with a single model summary, as used in the early editions of the Document Understanding Conferences (Over et al., 2007).

In our work, we take the idea of automatic evaluation to an extreme and explore the feasibility of developing *a fully automatic* evaluation method for content selection that does not make use of human model summaries at all. To this end, we show that evaluating summaries by comparing them with the input obtains good correlations with manual evaluations for both query focused and update summarization tasks.

Our results have important implications for future development of summarization systems and their evaluation.

High correlations between system ranking produced with the fully automatic method and manual evaluations show that the new evaluation measures can be used during system development when human model summaries are not available.

Our results provide validation of several features that can be optimized in the development of new summarization systems when the objective is to improve content selection on average, over *a collection of test inputs*. However, none of the features is consistently predictive of good summary content for *individual inputs*.

We find that content selection performance on standard test collections can be approximated well by the proposed fully automatic method. This result greatly underlines the need to require linguistic quality evaluations alongside content selection ones in future evaluations and research.

2 Model-free methods for evaluation

Proposals for developing fully automatic methods for summary evaluation have been put forward in the past. Their attractiveness is obvious for large scale evaluations, or for evaluation on non-standard test sets for which human models are not available.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 306–314,
Singapore, 6-7 August 2009.

For example in Radev et al. (2003), a large scale fully automatic evaluation of eight summarization systems on *18,000 documents* was performed without any human effort. A search engine was used to rank documents according to their relevance to a given query. The summaries for each document were also ranked for relevance with respect to the same query. For good summarization systems, the relevance ranking of summaries is expected to be similar to that of the full documents. Based on this intuition, the correlation between relevance rankings of summaries and original documents was used to compare the different systems. The approach was motivated by the assumption that the distribution of terms in a good summary is similar to the distribution of terms in the original document.

Even earlier, Donaway et al. (2000) suggested that there are considerable benefits to be had in adopting model-free methods of evaluation involving direct comparisons between the original document and its summary. The motivation for their work was the considerable variation in content selection choices in model summaries (Rath et al., 1961). The identity of the model writer significantly affects summary evaluations (also noted by McKeown et al. (2001), Jing et al. (1998)) and evaluations of the same systems can be rather different when different models are used. In their experiments, Donaway et al. (2000) demonstrated that the correlations between manual evaluation using a model summary and

a) manual evaluation using a different model summary

b) automatic evaluation by directly comparing input and summary[1],

are the same. Their conclusion was that such automatic methods should be seriously considered as an alternative to model based evaluation.

In this paper, we present a comprehensive study of fully automatic summary evaluation without any human models. A summary's content is judged for quality by directly estimating its closeness to the input. We compare several probabilistic and information-theoretic approaches for characterizing the similarity and differences between input and summary content. A simple information-theoretic measure, Jensen Shannon divergence between input and summary, emerges as the best feature. System rankings produced using this measure lead to correlations as high as 0.88 with human judgements.

[1]They used cosine similarity to perform the input-summary comparison.

3 TAC summarization track

3.1 Query-focused and Update Summaries

Two types of summaries, query-focused and update summaries, were evaluated in the summarization track of the 2008 Text Analysis Conference (TAC)[2]. Query-focused summaries were produced from input documents in response to a stated user information need. The update summaries require more sophistication: two sets of articles on the same topic are provided. The first set of articles represents the background of a story and users are assumed to be already familiar with the information contained in them. The update task is to produce a multi-document summary from the second set of articles that can serve as an update to the user. This task is reminiscent of the novelty detection task explored at TREC (Soboroff and Harman, 2005).

3.2 Data

The test set for the TAC 2008 summarization task contains 48 inputs. Each input consists of two sets of 10 documents each, called docsets A and B. Both A and B are on the same general topic but B contains documents published later than those in A. In addition, the user's information need associated with each input is given by a query statement consisting of a title and narrative. An example query statement is shown below.

Title: Airbus A380

Narrative: Describe developments in the production and launch of the Airbus A380.

A system must produce two summaries: (1) a query-focused summary of docset A, (2) a compilation of updates from docset B, assuming that the user has read all the documents in A. The maximum length for both types of summaries is 100 words.

There were 57 participating systems in TAC 2008. We use the summaries and evaluations of these systems for the experiments reported in the paper.

3.3 Evaluation metrics

Both manual and automatic evaluations were conducted at NIST to assess the quality of summaries

[2]http://www.nist.gov/tac

manual score	R-1 recall	R-2 recall
Query Focused summaries		
pyramid score	0.859	0.905
responsiveness	0.806	0.873
Update summaries		
pyramid score	0.912	0.941
responsiveness	0.865	0.884

Table 1: Spearman correlation between manual scores and ROUGE-1 and ROUGE-2 recall. All correlations are highly significant with p-value $<$ 0.00001.

produced by the systems.

Pyramid evaluation: The pyramid evaluation method (Nenkova and Passonneau, 2004) has been developed for reliable and diagnostic assessment of content selection quality in summarization and has been used in several large scale evaluations (Nenkova et al., 2007). It uses multiple human models from which annotators identify semantically defined Summary Content Units (SCU). Each SCU is assigned a weight equal to the number of human model summaries that express that SCU. An ideal maximally informative summary would express a subset of the most highly weighted SCUs, with multiple maximally informative summaries being possible. The pyramid score for a system summary is equal to the ratio between the sum of weights of SCUs expressed in a summary (again identified manually) and the sum of weights of an ideal summary with the same number of SCUs.

Four human summaries provided by NIST for each input and task were used for the pyramid evaluation at TAC.

Responsiveness evaluation: Responsiveness of a summary is a measure of overall quality combining both content selection and linguistic quality: summaries must present useful content in a structured fashion in order to better satisfy the user's need. Assessors directly assigned scores on a scale of 1 (poor summary) to 5 (very good summary) to each summary. These assessments are done without reference to any model summaries. The (Spearman) correlation between the pyramid and responsiveness metrics is high but not perfect: 0.88 and 0.92 respectively for query focused and update summarization.

ROUGE evaluation: NIST also evaluated the summaries automatically using ROUGE (Lin, 2004; Lin and Hovy, 2003). Comparison between a summary and the set of four model summaries is computed using unigram (R1) and bigram overlaps (R2)[3]. The correlations between ROUGE and manual evaluations is shown in Table 1 and varies between 0.80 and 0.94.

Linguistic quality evaluation: Assessors scored summaries on a scale from 1 (very poor) to 5 (very good) for five factors of linguistic quality: grammaticality, non-redundancy, referential clarity, focus, structure and coherence.

We do not make use of any of the linguistic quality evaluations. Our work focuses on fully automatic evaluation of content selection, so manual pyramid and responsiveness scores are used for comparison with our automatic method. The pyramid metric measures content selection exclusively, while responsiveness incorporates at least some aspects of linguistic quality.

4 Features for content evaluation

We describe three classes of features to compare input and summary content: distributional similarity, summary likelihood and use of topic signatures. Both input and summary words were stopword filtered and stemmed before computing the features.

4.1 Distributional Similarity

Measures of similarity between two probability distributions are a natural choice for the task at hand. One would expect good summaries to be characterized by low divergence between probability distributions of words in the input and summary, and by high similarity with the input.

We experimented with three common measures: KL and Jensen Shannon divergence and cosine similarity. These three metrics have already been applied for summary evaluation, albeit in different contexts. In Lin et al. (2006), KL and JS divergences between human and machine summary distributions were used to evaluate content selection. The study found that JS divergence always outperformed KL divergence. Moreover, the performance of JS divergence was better than standard ROUGE scores for multi-document summarization when multiple human models were used for the comparison.

The use of cosine similarity in Donaway et al. (2000) is more directly related to our work. They show that the difference between evaluations

[3]The scores were computed after stemming but stop words were retained in the summaries.

based on two different human models is about the same as the difference between system ranking based on one model summary and the ranking produced using input-summary similarity. Inputs and summaries were compared using only one metric: cosine similarity.

Kullback Leibler (KL) divergence: The KL divergence between two probability distributions P and Q is given by

$$D(P||Q) = \sum_w p_P(w) \log_2 \frac{p_P(w)}{p_Q(w)} \quad (1)$$

It is defined as the average number of bits wasted by coding samples belonging to P using another distribution Q, an approximate of P. In our case, the two distributions are those for words in the input and summary respectively. Since KL divergence is not symmetric, both input-summary and summary-input divergences are used as features. In addition, the divergence is undefined when $p_P(w) > 0$ but $p_Q(w) = 0$. We perform simple smoothing to overcome the problem.

$$p(w) = \frac{C + \delta}{N + \delta * B} \quad (2)$$

Here C is the count of word w and N is the number of tokens; $B = 1.5|V|$, where V is the input vocabulary and δ was set to a small value of 0.0005 to avoid shifting too much probability mass to unseen events.

Jensen Shannon (JS) divergence: The JS divergence incorporates the idea that the distance between two distributions cannot be very different from the average of distances from their mean distribution. It is formally defined as

$$J(P||Q) = \frac{1}{2}[D(P||A) + D(Q||A)], \quad (3)$$

where $A = \frac{P+Q}{2}$ is the mean distribution of P and Q. In contrast to KL divergence, the JS distance is symmetric and always defined. We use both smoothed and unsmoothed versions of the divergence as features.

Similarity between input and summary: The third metric is cosine overlap between the $tf * idf$ vector representations (with max-tf normalization) of input and summary contents.

$$cos\theta = \frac{v_{inp}.v_{summ}}{||v_{inp}||||v_{summ}||} \quad (4)$$

We compute two variants:

1. Vectors contain all words from input and summary
2. Vectors contain only topic signatures from the input and all words of the summary

Topic signatures are words highly descriptive of the input, as determined by the application of log-likelihood test (Lin and Hovy, 2000). Using only topic signatures from the input to represent text is expected to be more accurate because the reduced vector has fewer dimensions compared with using all the words from the input.

4.2 Summary likelihood

The likelihood of a word appearing in the summary is approximated as being equal to its probability in the input. We compute both a summary's unigram probability as well as its probability under a multinomial model.

Unigram summary probability:

$$(p_{inp}w_1)^{n_1}(p_{inp}w_2)^{n_2}...(p_{inp}w_r)^{n_r} \quad (5)$$

where $p_{inp}w_i$ is the probability in the input of word w_i, n_i is the number of times w_i appears in the summary, and $w_1...w_r$ are all words in the summary vocabulary.

Multinomial summary probability:

$$\frac{N!}{n_1!n_2!...n_r!}(p_{inp}w_1)^{n_1}(p_{inp}w_2)^{n_2}...(p_{inp}w_r)^{n_r} \quad (6)$$

where $N = n_1 + n_2 + ... + n_r$ is the total number of words in the summary.

4.3 Use of topic words in the summary

Summarization systems that directly optimize for more topic signatures during content selection have fared very well in evaluations (Conroy et al., 2006). Hence the number of topic signatures from the input present in a summary might be a good indicator of summary content quality. We experiment with two features that quantify the presence of topic signatures in a summary:

1. Fraction of the summary composed of input's topic signatures.
2. Percentage of topic signatures from the input that also appear in the summary.

While both features will obtain higher values for summaries containing many topic words, the first is guided simply by the presence of any topic word while the second measures the diversity of topic words used in the summary.

4.4 Feature combination using linear regression

We also evaluated the performance of a linear regression metric combining all of the above features. The value of the regression-based score for each summary was obtained using a leave-one-out approach. For a particular input and system-summary combination, the training set consisted only of examples which included neither the same input nor the same system. Hence during training, no examples of either the test input or system were seen.

5 Correlations with manual evaluations

In this section, we report the correlations between system ranking using our automatic features and the manual evaluations. We studied the predictive power of features in two scenarios.

MACRO LEVEL; PER SYSTEM: The values of features were computed for each summary submitted for evaluation. For each system, the feature values were averaged across all inputs. All participating systems were ranked based on the average value. Similarly, the average manual score, pyramid or responsiveness, was also computed for each system. The correlations between the two rankings are shown in Tables 2 and 4.

MICRO LEVEL; PER INPUT: The systems were ranked for each input separately, and correlations between the summary rankings for each input were computed (Table 3).

The two levels of analysis address different questions: *Can we automatically identify system performance across all test inputs (macro level) and can we identify which summaries for a given input were good and which were bad (micro level)?* For the first task, the answer is a definite "yes" while for the second task the results are mixed.

In addition, we compare our results to model-based evaluations using ROUGE and analyze the effects of stemming the input and summary vocabularies. In order to allow for in-depth discussion, we will analyze our findings only for query focused summaries. Similar results were obtained for the evaluation of update summaries and are described in Section 7.

5.1 Performance at macro level

Table 2 shows the Spearman correlation between manual and automatic scores averaged across the 48 inputs. We find that both distributional similarity and the topic signature features produce system rankings very similar to those produced by humans. Summary probabilities, on the other hand, turn out to be unpredictive of content selection performance. The linear regression combination of features obtains high correlations with manual scores but does not lead to better results than the single best feature: JS divergence.

Features	pyramid	respons.
JS div	-0.880	-0.736
JS div smoothed	-0.874	-0.737
% of input topic words	0.795	0.627
KL div summ-inp	-0.763	-0.694
cosine overlap	0.712	0.647
% of summ = topic wd	0.712	0.602
topic overlap	0.699	0.629
KL div inp-summ	-0.688	-0.585
mult. summary prob.	0.222	0.235
unigram summary prob.	-0.188	-0.101
regression	0.867	0.705
ROUGE-1 recall	0.859	0.806
ROUGE-2 recall	0.905	0.873

Table 2: Spearman correlation on macro level for the query focused task. All results are highly significant with p-values < 0.000001 except unigram and multinomial summary probability, which are not significant even at the 0.05 level.

JS divergence outperforms other features including the regression metric and obtains the best correlations with both types of manual scores, 0.88 with pyramid score and 0.74 with responsiveness. The regression metric performs comparably with correlations of 0.86 and 0.70. The correlations obtained by both JS divergence and the regression metric with pyramid evaluations are in fact better than that obtained by ROUGE-1 recall (0.85).

The best topic signature based feature—percentage of input's topic signatures that are present in the summary—ranks next only to JS divergence and regression. The correlation between this feature and pyramid and responsiveness evaluations is 0.79 and 0.62 respectively. The proportion of summary content composed of topic words performs worse as an evaluation metric with correlations 0.71 and 0.60. This result indicates that summaries that cover more topics from the input are judged to have better content than those in which fewer topics are mentioned.

Cosine overlaps and KL divergences obtain good correlations but still lower than JS divergence or percentage of input topic words. Further, rankings based on unigram and multinomial sum-

mary probabilities do not correlate significantly with manual scores.

5.2 Performance on micro level

On a per input basis, the proposed metrics are not that effective in distinguishing which summaries have better content. The minimum and maximum correlations with manual evaluations across the 48 inputs are given in Table 3. The number and percentage of inputs for which correlations were significant are also reported.

Now, JS divergence obtains significant correlations with pyramid scores for 73% of the inputs but for particular inputs, the correlation can be as low as 0.27. The results are worse for other features and for comparison with responsiveness scores.

At the micro level, combining features with regression gives the best result overall, in contrast to the findings for the macro level setting. This result has implications for system development; no single feature can reliably predict good content for a particular input. Even a regression combination of all features is a significant predictor of content selection quality in only 77% of the cases.

We should note however, that our features are based only on the distribution of terms in the input and therefore less likely to inform good content for all input types. For example, a set of documents each describing different opinion on a given issue will likely have less repetition on both lexical and content unit level. The predictiveness of features like ours will be limited for such inputs[4]. However, model summaries written for the specific input would give better indication of what information in the input was important and interesting. This indeed is the case as we shall see in Section 6.

Overall, the micro level results suggest that the fully automatic measures we examined will not be useful for providing information about summary quality for an individual input. For averages over many test sets, the fully automatic evaluations give more reliable and useful results, highly correlated with rankings produced by manual evaluations.

[4]In fact, it would be surprising to find an automatically computable feature or feature combination which would be able to consistently predict good content for all individual inputs. If such features existed, an ideal summarization system would already exist.

5.3 Effects of stemming

The analysis presented so far is on features computed after stemming the input and summary words. We also computed the values of the same features without stemming and found that divergence metrics benefit greatly when stemming is done. The biggest improvements in correlations are for JS and KL divergences with respect to responsiveness. For JS divergence, the correlation increases from 0.57 to 0.73 and for KL divergence (summary-input), from 0.52 to 0.69.

Before stemming, the topic signature and bag of words overlap features are the best predictors of responsiveness (correlations are 0.63 and 0.64 respectively) but do not change much after stemming (topic overlap—0.62, bag of words—0.64). Divergences emerge as better metrics only after stemming.

Stemming also proves beneficial for the likelihood features. Before stemming, their correlations are directed in the wrong direction, but they improve after stemming to being either positive or closer to zero. However, even after stemming, summary probabilities are not good predictors of content quality.

5.4 Difference in correlations: pyramid and responsiveness scores

Overall, we find that correlations with pyramid scores are higher than correlations with responsiveness. Clearly our features are designed to compare input-summary content only. Since responsiveness judgements were based on both content and linguistic quality of summaries, it is not surprising that these rankings are harder to replicate using our content based features. Nevertheless, responsiveness scores are dominated by content quality and the correlation between responsiveness and JS divergence is high, 0.73.

Clearly, metrics of linguistic quality should be integrated with content evaluations to allow for better predictions of responsiveness. To date, few attempts have been made to automatically evaluate linguistic quality in summarization. Lapata and Barzilay (2005) proposed a method for coherence evaluation which holds promise but has not been validated so far on large datasets such as those used in TAC and DUC. In a simpler approach, Conroy and Dang (2008) use higher order ROUGE scores to approximate both content and linguistic quality.

	pyramid			responsiveness		
features	**max**	**min**	**no. significant (%)**	**max**	**min**	**no. significant (%)**
JS div	-0.714	-0.271	35 (72.9)	-0.654	-0.262	35 (72.9)
JS div smoothed	-0.712	-0.269	35 (72.9)	-0.649	-0.279	33 (68.8)
KL div summ-inp	-0.736	-0.276	35 (72.9)	-0.628	-0.261	35 (72.9)
% of input topic words	0.701	0.286	31 (64.6)	0.693	0.279	29 (60.4)
cosine overlap	0.622	0.276	31 (64.6)	0.618	0.265	28 (58.3)
KL div inp-summ	-0.628	-0.262	28 (58.3)	-0.577	-0.267	22 (45.8)
topic overlap	0.597	0.265	30 (62.5)	0.689	0.277	26 (54.2)
% summary = topic wd	0.607	0.269	23 (47.9)	0.534	0.272	23 (47.9)
mult. summary prob.	0.434	0.268	8 (16.7)	0.459	0.272	10 (20.8)
unigram summary prob.	0.292	0.261	2 (4.2)	0.466	0.287	2 (4.2)
regression	0.736	0.281	37 (77.1)	0.642	0.262	32 (66.7)
ROUGE-1 recall	0.833	0.264	47 (97.9)	0.754	0.266	46 (95.8)
ROUGE-2 recall	0.875	0.316	48 (100)	0.742	0.299	44 (91.7)

Table 3: Spearman correlations at micro level (query focused task). Only the minimum, maximum values of the significant correlations are reported together with the number and percentage of significant correlations.

	update input only		avg. update & background	
features	**pyramid**	**respons.**	**pyramid**	**respons.**
JS div	-0.827	-0.764	-0.716	-0.669
JS div smoothed	-0.825	-0.764	-0.713	-0.670
% of input topic words	0.770	0.709	0.677	0.616
KL div summ-inp	-0.749	-0.709	-0.651	-0.624
KL div inp-summ	-0.741	-0.717	-0.644	-0.638
cosine overlap	0.727	0.691	0.649	0.631
% of summary = topic wd	0.721	0.707	0.647	0.636
topic overlap	0.707	0.674	0.645	0.619
mult. summmary prob.	0.284	0.355	0.152	0.224
unigram summary prob.	-0.093	0.038	-0.151	-0.053
regression	0.789	0.605	0.699	0.522
ROUGE-1 recall	0.912	0.865	.	.
ROUGE-2 recall	0.941	0.884	.	.

regression combining features comparing with background and update inputs (without averaging)

correlations = 0.8058 with pyramid, 0.6729 with responsiveness

Table 4: Spearman correlations at macro level for update summarization. Results are reported separately for features comparing update summaries with the update input only or with both update and background inputs and averaging the two.

6 Comparison with ROUGE

For manual pyramid scores, the best correlation, 0.88, we observed in our experiments was with JS divergence. This result is unexpectedly high for a fully automatic evaluation metric. Note that the best correlation between pyramid scores and ROUGE (for R2) is 0.90, practically identical with JS divergence. For ROUGE-1, the correlation is 0.85.

In the case of manual responsiveness, which combines aspects of linguistic quality along with content selection evaluation, the correlation with JS divergence is 0.73. For ROUGE, it is 0.80 for R1 and 0.87 for R2. Using higher order n-grams is obviously beneficial as observed from the differences between unigram and bigram ROUGE scores. So a natural extension of our features would be to use distance between bigram distributions. At the same time, for responsiveness, ROUGE-1 outperforms all the fully automatic features. This is evidence that the model summaries provide information that is unlikely to ever be approximated by information from the input alone, regardless of feature sophistication.

At the micro level, ROUGE does clearly better than all the automatic measures. The results are shown in the last two rows of Table 3. ROUGE-1 recall obtains significant correlations for over 95% of inputs for responsiveness and 98% of inputs for pyramid evaluation compared to 73% (JS divergence) and 77% (regression). Undoubtedly, at the input level, comparison with model summaries is substantially more informative.

When reference summaries are available, ROUGE provides scores that agree best with human judgements. However, when model sum-

maries are not available, our features can provide reliable estimates of system quality when averaged over a set of test inputs. For predictions at the level of individual inputs, our fully automatic features are less useful.

7 Update Summarization

In Table 4, we report the performance of our features for system evaluation on the update task. The column, "update input only" summarizes the correlations obtained by features comparing the summaries with only the update inputs (set B). We also compared the summaries individually to the update and background (set A) inputs. The two sets of features were then combined by a) averaging ("avg. update and background") and b) linear regression (last line of Table 4).

As in the case of query focused summarization, JS divergence and percentage of input topic signatures in summary are the best features for the update task as well. The overall best feature is JS divergence between the update input and the summaries—correlations of 0.82 and 0.76 with pyramid and responsiveness.

Interestingly, the features combining both update and background inputs do not lead to better correlations than those obtained using the update input only. The best performance from combined features is given by the linear regression metric. Although the correlation of this regression feature with pyramid scores (0.80) is comparable to JS divergence with update inputs, its correlation with responsiveness (0.67) is clearly lower. These results show that the term distributions in the update input are sufficiently good predictors of content for update summaries. The role of the background input appears to be negligable.

8 Discussion

We have presented a successful framework for model-free evaluations of content which uses the input as reference. The power of model-free evaluations generalizes across at least two summarization tasks: query focused and update summarization.

We have analyzed a variety of features for input-summary comparison and demonstrated that the strength of different features varies considerably. Similar term distributions in the input and the summary and diverse use of topic signatures in the summary are highly indicative of good content. We also find that preprocessing like stemming improves the performance of KL and JS divergence features.

Very good results were obtained from a correlation analysis with human judgements, showing that input can indeed substitute for model summaries and manual efforts in summary evaluation. The best correlations were obtained by a single feature, JS divergence (0.88 with pyramid scores and 0.73 with responsiveness at system level).

Our best features can therefore be used to evaluate the content selection performance of systems in a new domain where model summaries are unavailable. However, like all other content evaluation metrics, our features must be accompanied by judgements of linguistic quality to obtain wholesome indicators of summary quality and system performance. Evidence for this need is provided by the lower correlations with responsiveness than the content-only pyramid evaluations.

The results of our analysis zero in on JS divergence and topic signature as desirable objectives to optimize during content selection. On the macro level, they are powerful predictors of content quality. These findings again emphasize the need for always including linguistic quality as a component of evaluation.

Observations from our input-based evaluation also have important implications for the design of novel summarization tasks. We find that high correlations with manual evaluations are obtained by comparing query-focused summaries with the entire input and making no use of the query at all. Similarly in the update summarization task, the best predictions of content for update summaries were obtained using only the update input. The uncertain role of background inputs and queries expose possible problems with the task designs. Under such conditions, it is not clear if query-focused content selection or ability to compile updates are appropriately captured by any evaluation.

References

J. Conroy and H. Dang. 2008. Mind the gap: Dangers of divorcing evaluations of summary content from linguistic quality. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 145–152.

J. Conroy, J. Schlesinger, and D. O'Leary. 2006. Topic-focused multi-document summarization using an approximate oracle score. In *Proceedings of ACL, short paper*.

R. Donaway, K. Drummey, and L. Mather. 2000. A comparison of rankings produced by summarization evaluation measures. In *NAACL-ANLP Workshop on Automatic Summarization*.

H. Jing, R. Barzilay, K. Mckeown, and M. Elhadad. 1998. Summarization evaluation methods: Experiments and analysis. In *In AAAI Symposium on Intelligent Summarization*, pages 60–68.

M. Lapata and R. Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *IJCAI'05*.

C. Lin and E. Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th conference on Computational linguistics*, pages 495–501.

C. Lin and E. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurance statistics. In *Proceedings of HLT-NAACL 2003*.

C. Lin, G. Cao, J. Gao, and J. Nie. 2006. An information-theoretic approach to automatic evaluation of summaries. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 463–470.

C. Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *ACL Text Summarization Workshop*.

K. McKeown, R. Barzilay, D. Evans, V. Hatzivassiloglou, B. Schiffman, and S. Teufel. 2001. Columbia multi-document summarization: Approach and evaluation. In *DUC'01*.

A. Nenkova and R. Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *HLT/NAACL*.

A. Nenkova, R. Passonneau, and K. McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Trans. Speech Lang. Process.*, 4(2):4.

P. Over, H. Dang, and D. Harman. 2007. Duc in context. *Inf. Process. Manage.*, 43(6):1506–1520.

D. Radev, S. Teufel, H. Saggion, W. Lam, J. Blitzer, H. Qi, A. Çelebi, D. Liu, and E. Drabek. 2003. Evaluation challenges in large-scale multi-document summarization: the mead project. In *Proceedings of ACL 2003*, Sapporo, Japan.

G. J. Rath, A. Resnick, and R. Savage. 1961. The formation of abstracts by the selection of sentences: Part 1: sentence selection by man and machines. *American Documentation*, 2(12):139–208.

I. Soboroff and D. Harman. 2005. Novelty detection: the trec experience. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 105–112.

Classifier Combination for Contextual Idiom Detection Without Labelled Data

Linlin Li and **Caroline Sporleder**
Saarland University
Postfach 15 11 50
66041 Saarbrücken
Germany
{linlin,csporled}@coli.uni-saarland.de

Abstract

We propose a novel unsupervised approach for distinguishing literal and non-literal use of idiomatic expressions. Our model combines an unsupervised and a supervised classifier. The former bases its decision on the cohesive structure of the context and labels training data for the latter, which can then take a larger feature space into account. We show that a combination of both classifiers leads to significant improvements over using the unsupervised classifier alone.

1 Introduction

Idiomatic expressions are abundant in natural language. They also often behave idiosyncratically and are therefore a significant challenge for natural language processing systems. For example, idioms can violate selectional restrictions (as in *push one's luck*), disobey typical subcategorisation constraints (e.g., *in line* without a determiner before *line*), or change the default assignments of semantic roles to syntactic categories (e.g., in *break sth with X* the argument *X* would typically be an instrument but for the idiom *break the ice* it is more likely to fill a patient role, as in *break the ice with Russia*).

In order to deal with such idiosyncracies and assign the correct analyses, NLP systems need to be able to recognise idiomatic expressions. Much previous research on idioms has been concerned with *type-based classification*, i.e., dividing expressions into 'idiom' or 'not idiom' irrespective of their actual use in a given context. However, while some expressions, such as *by and large*, always have an idiomatic meaning, several other expressions, such as *break the ice* or *spill the beans*, can be used literally as well as idiomatically (see examples (1) and (2), respectively). Sometimes the literal usage can even dominate in a domain, as for *drop the ball*, which occurs fairly frequently in a literal sense in the sports section of news texts.

(1) Dad had to break the ice on the chicken troughs so that they could get water.

(2) Somehow I always end up spilling the beans all over the floor and looking foolish when the clerk comes to sweep them up.

Hence, whether a particular occurrence of a potentially ambiguous expression has literal or non-literal meaning has to be inferred from the context (*token-based idiom classification*). Recently, there has been increasing interest in this classification task and both supervised and unsupervised techniques have been proposed. The work we present here builds on previous research by Sporleder and Li (2009), who describe an unsupervised method that exploits the presence or absence of cohesive ties between the component words of a potential idiom and its context to distinguish between literal and non-literal use. If strong ties can be found the expression is classified as literal otherwise as non-literal. While this approach often works fairly well, it has the disadvantage that it focuses exclusively on lexical cohesion, other linguistic cues that might influence the classification decision are disregarded.

We show that it is possible to improve on Sporleder and Li's (2009) results by employing a two-level strategy, in which a cohesion-based unsupervised classifier is combined with a supervised classifier. We use the unsupervised classifier to label a sub-set of the test data with high confidence. This sub-set is then passed on as training data to the supervised classifier, which then labels the remainder of the data set. Compared to a fully unsupervised approach, this two-stage method has the advantage that a larger feature set can be exploited. This is beneficial for examples, in which the cohesive ties are relatively weak but which contain other linguistic cues for literal or non-literal use.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 315–323,
Singapore, 6-7 August 2009.

2 Related Work

Most studies on idiom classification focus on type-based classification; few researchers have worked on token-based approaches (i.e., classification of an expression in a given context). Type-based methods frequently exploit the fact that idioms have a number of properties which differentiate them from other expressions. For example, they often exhibit a degree of syntactic and lexical fixedness. Some idioms, for instance, do not allow internal modifiers (**shoot the long breeze*) or passivisation (**the bucket was kicked*). They also typically only allow very limited lexical variation (**kick the vessel*, **strike the bucket*).

Many approaches for identifying idioms focus on one of these two aspects. For instance, measures that compute the association strength between the elements of an expression have been employed to determine its degree of compositionality (Lin, 1999; Fazly and Stevenson, 2006) (see also Villavicencio et al. (2007) for an overview and a comparison of different measures). Other approaches use Latent Semantic Analysis (LSA) to determine the similarity between a potential idiom and its components (Baldwin et al., 2003). Low similarity is supposed to indicate low compositionality. Bannard (2007) looks at the syntactic fixedness of idiomatic expressions, i.e., how likely they are to take modifiers or be passivised, and compares this to what would be expected based on the observed behaviour of the component words. Fazly and Stevenson (2006) combine information about syntactic and lexical fixedness (i.e., estimated degree of compositionality) into one measure.

The few token-based approaches include a study by Katz and Giesbrecht (2006), who devise a supervised method in which they compute the meaning vectors for the literal and non-literal usages of a given expression in the training data. An unseen test instance of the same expression is then labelled by performing a nearest neighbour classification.

Birke and Sarkar (2006) model literal vs. non-literal classification as a word sense disambiguation task and use a clustering algorithm which compares test instances to two automatically constructed seed sets (one with literal and one with non-literal expressions), assigning the label of the closest set. While the seed sets are created without immediate human intervention they do rely on manually created resources such as databases of known idioms.

Cook et al. (2007) and Fazly et al. (2009) propose an alternative method which crucially relies on the concept of *canonical form*, which is a fixed form (or a small set of those) corresponding to the syntactic pattern(s) in which the idiom normally occurs (Riehemann, 2001).[1] The canonical form allows for inflectional variation of the head verb but not for other variations (such as nominal inflection, choice of determiner etc.). It has been observed that if an expression is used idiomatically, it typically occurs in its canonical form. For example, Riehemann (2001, p. 34) found that for decomposable idioms 75% of the occurrences are in canonical form, rising to 97% for non-decomposable idioms.[2] Cook et al. exploit this behaviour and propose an unsupervised method which classifies an expression as idiomatic if it occurs in canonical form and literal otherwise.

Finally, in earlier work, we proposed an unsupervised method which detects the presence or absence of cohesive links between the component words of the idiom and the surrounding discourse (Sporleder and Li, 2009). If such links can be found the expression is classified as 'literal' otherwise as 'non-literal'. In this paper we show that the performance of such a classifier can be significantly improved by complementing it with a second-stage supervised classifier.

3 First Stage: Unsupervised Classifier

As our first-stage classifier, we use the unsupervised model proposed by Sporleder and Li (2009). This model exploits the fact that words in a coherent discourse exhibit *lexical cohesion* (Halliday and Hasan, 1976), i.e. concepts referred to in sentences are typically related to other concepts mentioned elsewhere in the discourse. Given a suitable measure of semantic relatedness, it is possible to compute the strength of such cohesive ties between pairs of words. While the component words of literally used expressions tend to exhibit lexical cohesion with their context, the words of non-literally used expressions do not. For example, in (3) the expression *play with fire* is used literally and the word *fire* is related to surrounding words like *grilling*, *dry-heat*, *cooking*, and *coals*. In (4), however *play with fire* is used non-literally and cohesive ties be-

[1] This is also the form in which an idiom is usually listed in a dictionary.

[2] Decomposable idioms are expressions such as *spill the beans* which have a composite meaning whose parts can be mapped to the words of the expression (e.g., *spill*→'reveal', *beans*→'secret').

tween *play* or *fire* and the context are absent.

(3) **Grilling** outdoors is much more than just another **dry-heat cooking** method. It's the chance to play with **fire**, satisfying a primal urge to stir around in **coals** .

(4) And PLO chairman Yasser Arafat has accused Israel of playing with fire by supporting HAMAS in its infancy.

To determine the strength of cohesive links, the unsupervised model builds a graph structure (called *cohesion graph*) in which all pairs of content words in the context are connected by an edge which is weighted by the pair's semantic relatedness. Then the *connectivity* of the graph is computed, defined as the average edge weight. If the connectivity increases when the component words of the idiom are removed, then there are no strong cohesive ties between the expression and the context and the example is labelled as 'non-literal', otherwise it is labelled as 'literal'.

To model semantic distance, we use the *Normalized Google Distance* (NGD, see Cilibrasi and Vitanyi (2007)), which computes relatedness on the basis of page counts returned by a search engine.[3] It is defined as follows:

$$NGD(x,y) = \frac{max\{log\ f(x), log\ f(y)\} - log\ f(x,y)}{log\ M - min\{log\ f(x), log\ f(y)\}} \quad (5)$$

where x and y are the two words whose association strength is computed (e.g., *fire* and *coal*), $f(x)$ is the page count returned by the search engine for x (and likewise for $f(y)$ and y), $f(x,y)$ is the page count returned when querying for "x AND y", and M is the number of web pages indexed by the search engine. The basic idea is that the more often two terms occur together, relative to their overall occurrence, the more closely they are related.

We hypothesise that the unsupervised classifier will give us relatively good results for some examples. For instance, in (3) there are several strong cues which suggest that *play with fire* is used literally. However, because the unsupervised classifier only looks at lexical cohesion, it misses many other clues which could help distinguish literal and non-literal usages. For example, if *break the ice* is followed by the prepositions *between* or *over* as in example (6), it is more likely to be used idiomatically (at least in the news domain).

(6) "Gujral will meet Sharif on Monday and discuss bilateral relations," the Press Trust of India added. The minister said Sharif and Gujral would be able to break the ice over Kashmir.

Furthermore, idiomatic usages also exhibit cohesion with their context but the cohesive ties are with the *non-literal* meaning of the expression. For example, in news texts, *break the ice* in its figurative meaning often co-occurs with *discuss*, *relations*, *talks* or *diplomacy* (see (6)). At the moment we do not have any way to model these cohesive links, as we do not know the non-literal meaning of the idiom.[4] However if we had labelled data we could train a supervised classifier to learn these and other contextual clues. The trained classifier might then be able to correctly classify examples which were misclassified by the unsupervised classifier, i.e., examples in which the cohesive ties are weak but where other clues exist which indicate how the expression is used.

For example, in (7) there is weak cohesive evidence for a literal use of *break the ice*, due to the semantic relatedness between *ice* and *water*. However, there are stronger cues for non-literal usage, such as the preposition *between* and the presence of words like *diplomats* and *talks*, which are indicative of idiomatic usage. Examples like this are likely to be misclassified by the unsupervised model; a supervised classifier, on the other hand, has a better chance to pick up on such additional cues and predict the correct label.

(7) Next week the two diplomats will meet in an attempt to break the ice between the two nations. A crucial issue in the talks will be the long-running water dispute.

4 Second Stage: Supervised Classifier

For the supervised classifier, we used Support Vector Machines as implemented by the LIBSVM package.[5] We implemented four types of features, which encode both cohesive information and word co-occurrence more generally.[6]

[3] We employ Yahoo! rather than Google since we found that it returns more stable counts.

[4] It might be possible to compute the Normalized Google Distance between the whole expression and the words in the context, assuming that whenever the whole expression occurs it is much more likely to be used figuratively than literally. For expressions in canonical form this is indeed often the case (Riehemann, 2001), however there are exceptions (see Section 6.1) for which such an approach would not work.

[5] Available from: `http://www.csie.ntu.edu.tw/~cjlin/libsvm/` We used the default parameters.

[6] We also experimented with linguistically more informed features, such as the presence of named entities in the local context of the expression, and properties of the subject or co-ordinated verbs, but we found that these features did not lead to a better performance of the supervised classifier. This is probably partly due to data sparseness.

Salient Words (salW) This feature aims to identify words which are particularly *salient* for literal usage. We used a frequency-based definition of salience and computed the *literal saliency score* for each word in a five-paragraph context around the target expression:

$$sal_{lit}(w) = \frac{\log f_{lit}(w) \times i_{lit}(w)}{\log f_{nonlit}(w) \times i_{nonlit}(w)} \quad (8)$$

where $sal_{lit}(w)$ is the saliency score of the word w for the class lit; $f_{lit}(w)$ is the token frequency of the word w for literally used expressions; $i_{lit}(w)$ is the number of instances of the target expressions classified as lit which co-occur with word w (and mutatis mutandis $nonlit$ for target expressions labelled as non-literal).[7]

Words with a high sal_{lit} occur much more frequently with literal usages than with non-literal ones. Conversely, words with a low sal_{lit} should be more indicative of the non-literal class. However, we found that, in practice, the measure is better at picking out indicative words for the literal class; non-literal usages tend to co-occur with a wide range of words. For example, among the highest scoring words for *break the ice* we find *thick, bucket, cold, water, reservoir* etc. While we do find words like *relations, diplomacy, discussions* among the lowest scoring terms (i.e., terms indicative of the non-literal class), we also find a lot of noise (*ask, month*). The effect is even more pronounced for other expressions (like *drop the ball*) which tend to be used idiomatically in a wider variety of situations (*drop the ball on a ban of chemical weapons*, *drop the ball on debt reduction* etc.).

We implement the saliency score in our model by encoding for the 300 highest scoring words whether the word is present in the context of a given example and how frequently it occurs.[8] Note that this feature (as well as the next one) can be computed in a per-idiom or a generic fashion. In the former case, we would encode the top 300 words separately for each idiom in the training set, in the latter across all idioms (with the consequence that more frequent idioms in the training set contribute to more positions in the feature vector). We found that, in practice, it does not make a big difference which variant is used. Moreover, in our bootstrapping scenario, we cannot ensure that we have sufficient examples of each idiom in the training set to train separate classifiers, so we opted for generic models throughout all experiments.

Related Words (relW) This feature set is a variant of the previous one. Here we score the words not based on their saliency but we determine the semantic relatedness between the noun in the idiomatic expression and each word in the global context, using the *Normalized Google Distance* mentioned in Section 3. Again we encode the 300 top-scoring words.

While the *related words* feature is less prone to overestimation of accidental co-occurrence than the saliency feature, it has the disadvantage of conflating different word senses. For example, among the highest scoring words for *ice* are *cold, melt, snow, skate, hockey* but also *cream, vanilla, dessert*.

Relatedness Score (relS) The fourth feature set implements the *relatedness score* which encodes the scores for the 100 most highly weighted edges in the cohesion graph of an instance.[9] If these scores are high, there are many cohesive ties with the surrounding discourse and the target expression is likely to be used literally.

Discourse Connectivity (connect.) Finally, we implemented two features which look at the cohesion graph of an instance. We encode the connectivity of the graph (i) when the target expression is included and (ii) when it is excluded. The unsupervised classifier uses the difference between these two values to make its prediction. By encoding the absolute connectivity values as features we enable the supervised classifier to make use of this information as well.

5 Combining the Classifiers

As mentioned before, we use the unsupervised classifier to label an initial training set for the supervised one. To ensure that the training set does not contain too much noise, we only add those examples about which the unsupervised classifier is

[7] Our definition of sal_{lit} bears similarities with the well known $tf.idf$ score. We include both the term frequencies (f_{lit}) and the instance frequencies (i_{lit}) in the formula because we believe both are important. However, the instance frequency is more informative and less sensitive to noise because it indicates that expression classified as 'literal' *consistently* co-occurs with the word in question. Therefore we weight down the effect of the term frequency by taking its log.

[8] We also experimented with different feature dimensions besides 300 but did not find a big difference in performance.

[9] We only used the 100 highest ranked edges because we are looking at a specific context here rather than the contexts of the literal or non-literal class overall. Since the contexts we use are only five paragraphs long, recording the 100 strongest edges seems sufficient.

most confident. We thus need to address two questions: (i) how to define a *confidence function* for the unsupervised classifier, and (ii) how to set the *confidence threshold* governing what proportion of the data set is used for training the second classifier.

The first question is relatively easy to answer: as the unsupervised classifier bases its decision on the difference in connectivity between including or excluding the component words of the idiom in the cohesion graph, an obvious choice for a confidence function is the difference in connectivity; i.e., the higher the difference, the higher the confidence of the classifier in the predicted label.

The confidence threshold could be selected on the basis of the unsupervised classifier's performance on a development set. Note that when choosing such a threshold there is usually a trade-off between the size of the training set and the amount of noise in it: the lower the threshold, the larger and the noisier the training set. Ideally we would like a reasonably-sized training set which is also relatively noise-free, i.e., does not contain too many wrongly labelled examples. One way to achieve this is to start with a relatively small training set and then expand it gradually.

A potential problem for the supervised classifier is that our data set is relatively imbalanced, with the non-literal class being four times as frequent as the literal class. Supervised classifiers often have problems with imbalanced data and tend to be overly biased towards the majority class (see, e.g., Japkowicz and Stephen (2002)). To overcome this problem, we experimented with boosting the literal class with additional examples.[10] We describe our methods for training set enlargement and boosting the literal class in the remainder of this section.

Iteratively Enlarging the Training Set A typical method for increasing the training set is to go through several iterations of enlargement and retraining.[11] We adopt a conservative enlargement strategy: we only consider instances on whose labels both classifiers agree and we use the confidence function of the unsupervised classifier to determine which of these examples to add to the training set. The motivation for this is that we hypothesise that the supervised classifier will not have a very good performance initially, as it is trained on a very small data set. As a consequence its confidence function may also not be very accurate. On the other hand, we know from Sporleder and Li (2009) that the unsupervised classifier has a reasonably good performance. So while we give the supervised classifier a veto-right, we do not allow it to select new training data by itself or overturn classifications made by the unsupervised classifier.

A similar strategy was employed by Ng and Cardie (2003) in a self-training set-up. However, while they use an ensemble of supervised classifiers, which they re-train after each iteration, we can only re-train the second classifier; the first one, being unsupervised, will never change its prediction. Hence it does not make sense to go through a large number of iterations; the more iterations we go through, the closer the performance of the combined classifier will be to that of the unsupervised one because that classifier will label a larger and larger proportion of the data. However, going through one or two iterations allows us to slowly enlarge the training set and thereby gradually improve the performance of the supervised classifier.

In each iteration, we select 10% of the remaining examples to be added to the training set.[12] We could simply add those 10% of the data about which the unsupervised classifier is most confident, but if the classifier was more confident about one class than about the other, we would risk obtaining a severely imbalanced training set. Hence, we decided to separate examples classified as 'literal' from those classified as 'non-literal' and add the top 10% from each set. Provided the automatic classification is reasonably accurate, this will ensure that the distribution of classes in the training set is roughly similar to that in the overall data set at least at the early stages of the bootstrapping.

Boosting the Literal Class As the process goes on, we are still likely to introduce more and more imbalance in the training set. This is due to the fact that the supervised classifier is likely to have some bias towards the majority class (and our experiments in Section 6.2 suggest that this is indeed the case). Hence, as the bootstrapping process goes on, potentially more and more examples will be labelled as 'non-literal' and if we always select the top 10% of these, our training set will gradually

[10] Throughout this paper, we use the term 'boosting' in a non-technical sense.

[11] In our case re-training also involves re-computing the ranked lists of salient and related words. As the process goes on the classifier will be able to discover more and more useful cue words and encode them in the feature vector.

[12] Since we do not have a separate development set, we chose the value of 10% intuitively as it seemed a reasonably good threshold.

become more imbalanced. This is a well-known problem for bootstrapping approaches (Blum and Mitchell, 1998; Le et al., 2006). We could counteract this by selecting a higher proportion of examples labelled as 'literal'. However given that the number of literal examples in our data set is relatively small, we would soon deplete our literal instance pool and moreover, because we would be forced to add less confidently labelled examples for the literal class, we are likely to introduce more noise in the training set.

A better option is to boost the literal class with external examples. To do this we exploit the fact that non-canonical forms of idioms are highly likely to be used literally. Given that our data set only contains canonical forms (see Section 6.1), we automatically extract non-canonical form variants and label them as 'literal'. To generate possible variants, we either (i) change the number of the noun (e.g., *rock the boat* becomes *rock the boats*), (ii) change the determiner (e.g., *rock a boat*), or (iii) replace the verb or noun by one of its synonyms, hypernyms, or siblings from WordNet (e.g., *rock the ship*). While this strategy does not give us additional literal examples for all idioms, for example we were not able to find non-canonical form occurrences of *sweep under the carpet* in the Gigaword corpus, for most idioms we were able to generate additional examples. Note that this data set is potentially noisy as not all non-canonical form examples are used literally. However, when checking a small sample manually, we found that only very small percentage ($<< 1\%$) was mis-labelled.

To reduce the classifier bias when enlarging the training set, we add additional literal examples during each iteration to ensure that the class distribution does not deviate too much from the distribution originally predicted by the unsupervised classifier.[13] The examples to be added are selected randomly but we try to ensure that each idiom is represented. When reporting the results, we disregard these additional external examples.

6 Experiments and Results

We carried out a number of different experiments. In Section 6.2 we investigate the performance of the different features of the supervised classifier and in Section 6.3 we look more closely at the behaviour of the combined classifier. We start by describing the data set.

6.1 Data

We used the data from Sporleder and Li (2009), which consist of 17 idioms that can be used both literally and non-literally (see Table 1). For each expression, all canonical form occurrences were extracted from the Gigaword corpus together with five paragraphs of context and labelled as 'literal' or 'non-literal'.[14] The inter-annotator agreement on a small sample of doubly annotated examples was 97% and the kappa score 0.7 (Cohen, 1960).

expression	literal	non-literal	all
back the wrong horse	0	25	25
bite off more than one can chew	2	142	144
bite one's tongue	16	150	166
blow one's own trumpet	0	9	9
bounce off the wall*	39	7	46
break the ice	20	521	541
drop the ball*	688	215	903
get one's feet wet	17	140	157
pass the buck	7	255	262
play with fire	34	532	566
pull the trigger*	11	4	15
rock the boat	8	470	478
set in stone	9	272	281
spill the beans	3	172	175
sweep under the carpet	0	9	9
swim against the tide	1	125	126
tear one's hair out	7	54	61
all	862	3102	3964

Table 1: Idiom statistics (* indicates expressions for which the literal usage is more common than the non-literal one)

6.2 Feature Analysis for the Supervised Classifier

In a first experiment, we tested the contribution of the different features (Table 2). For each set, we trained a separate classifier and tested it in 10-fold cross-validation mode. We also tested the performance of the first three features combined (salient and related words and relatedness score) as we wanted to know whether their combination leads to performance gains over the individual classifiers. Moreover, testing these three features in combination allows us to assess the contribution of the connectivity feature, which is most closely related to the unsupervised classifier. We report the accuracy, and because our data are fairly imbalanced,

[13] We are assuming that the true distribution is not known and use the predictions of the unsupervised classifier to approximate the true distribution.

[14] The restriction to canonical forms was motivated by the fact that for the mostly non-decomposable idioms in the set, the vast majority (97%) of non-canonical form occurrences will be used literally (see Section 2).

also the F-Score for the minority class ('literal').

Feature	Avg. literal (%) Prec.	Rec.	F-Score	Avg. (%) Acc.
salW	77.10	56.10	65.00	86.83
relW	78.00	43.20	55.60	84.99
relS	74.90	37.50	50.00	83.68
connectivity	78.30	2.10	4.10	78.58
salW+relW+relS	82.90	63.50	71.90	89.20
all	85.80	66.60	75.00	90.34

Table 2: Performance of different feature sets, 10-fold cross-validation

It can be seen that the *salient words* (*salW*) feature has the highest performance of the individual features, both in terms of accuracy and in terms of literal F-Score, followed by *related words* (*relW*), and *relatedness score* (*relS*). Intuitively, it is plausible that the saliency feature performs quite well as it can also pick up on linguistic indicators of idiom usage that do not have anything to do with lexical cohesion. However, a combination of the first three features leads to an even better performance, suggesting that the features do indeed model somewhat different aspects of the data.

The performance of the connectivity feature is also interesting: while it does not perform very well on its own, as it over-predicts the non-literal class, it noticeably increases the performance of the model when combined with the other features, suggesting that it picks up on complementary information.

6.3 Testing the Combined Classifier

We experimented with different variants of the combined classifier. The results are shown in Table 3. In particular, we looked at: (i) combining the two classifiers without training set enlargement or boosting of the literal class (*combined*), (ii) boosting the literal class with 200 automatically labelled non-canonical form examples (*combined+boost*), (iii) enlarging the training set by iteration (*combined+it*), and (iv) enlarging the training set by iteration and boosting the literal class after each iteration (*combined+boost+it*). The table shows the literal precision, recall and F-Score of the combined model (both classifiers) on the complete data set (excluding the extra literal examples). Note that the results for the set-ups involving iterative training set enlargement are optimistic: since we do not have a separate development set, we report the optimal performance achieved during the first seven iterations. In a real set-up, when the optimal number of iterations is chosen on the basis of a separate data set, the results may be lower. The table also shows the majority class baseline (*Base*$_{maj}$), and the overall performance of the unsupervised model (*unsup*) and the supervised model when trained in 10-fold cross-validation mode (*super 10CV*).

Model	Prec$_l$	Rec$_l$	F-Score$_l$	Acc.
Base$_{maj}$	-	-	-	78.25
unsup.	50.04	69.72	58.26	78.38
combined	83.86	45.82	59.26	86.30
combined+boost	70.26	62.76	66.30	86.13
combined+it*	85.68	46.52	60.30	86.68
combined+boost+it*	71.86	66.36	69.00	87.03
super. 10CV	85.80	66.60	75.00	90.34

Table 3: Results for different classifiers; $*$ indicates best performance (optimistic)

It can be seen that the combined classifier is 8% more accurate than both the majority baseline and the unsupervised classifier. This amounts to an error reduction of over 35% (the difference is statistically significant, χ^2 test, $p << 0.01$). While the F-Score of the unboosted combined classifier is comparable to that of the unsupervised one, boosting the literal class leads to a 7% increase, due to a significantly increased recall, with no significant drop in accuracy. These results show that complementing the unsupervised classifier with a supervised one, can lead to tangible performance gains. Note that the accuracy of the combined classifier, which uses no manually labelled training data, is only 4% below that of a fully supervised classifier; in other words, we do not lose much by starting with an automatically labelled data set. Iterative enlargement of the training set can lead to further improvements, especially when combined with boosting to reduce the classifier bias.

To get a better idea of the effect of training set enlargement, we plotted the accuracy and F-Score of the combined classifier for a given number of iterations with boosting (Figure 1) and without (Figure 2). It can be seen that enlargement has a noticeable positive effect if combined with boosting. If the literal class is not boosted, the increasing bias of the classifier seems to outweigh most of the positive effects from the enlarged training set. Figure 1 also shows that the best performance is obtained after a relatively small number of iterations (namely two), as expected.[15] With more iterations the performance decreases again. However, it decays rel-

[15] Note that this also depends on the confidence threshold. For example, if a threshold of 5% is chosen, more iterations may be required for optimal performance.

atively gracefully and even after seven iterations, when more than 40% of the data are classified by the unsupervised classifier, the combined classifier still achieves an overall performance that is significantly above that of the unsupervised classifier (84.28% accuracy compared to 78.38%, significant at $p << 0.01$). Hence, the combined classifier seems not to be very sensitive to the exact number of iterations and performs reasonably well even if the number of iterations is sub-optimal.

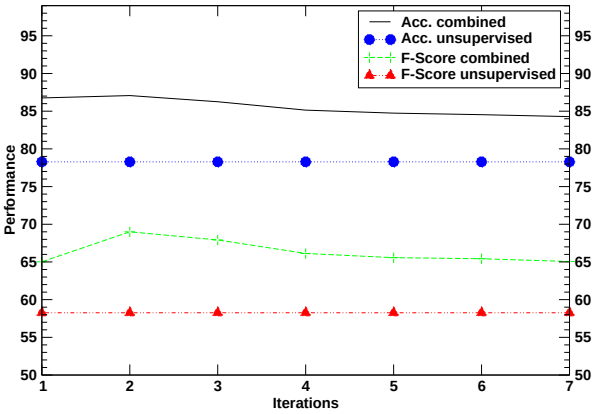


Figure 1: Accuracy and literal F-Score on complete data set after different iterations with boosting of the literal class

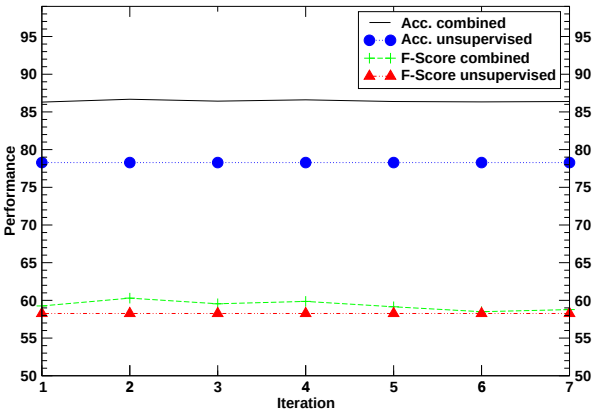


Figure 2: Accuracy and literal F-Score on complete data set after different iterations without boosting of the literal class

Figure 3 shows how the training set increases as the process goes on[16] and how the number of mis-classifications in the training set develops. Interestingly, when going from the first to the second iteration the training set nearly doubles (from 396 to 669 instances), while the proportion of errors is also reduced by a third (from 7% to 5%). Hence, the training set does not only grow but the proportion of noise in it decreases, too. This shows that our conservative enlargement strategy is fairly successful in selecting correctly labelled examples. Only at later stages, when the classifier bias takes over, does the proportion of noise increase again.

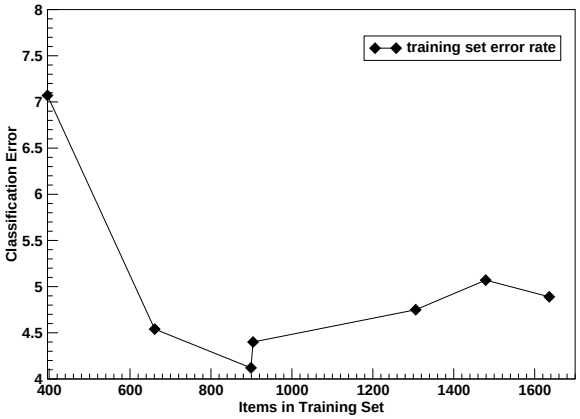


Figure 3: Training set size and error in training set at different iterations

7 Conclusion

We presented a two-stage classification approach for distinguishing literal and non-literal use of idiomatic expressions. Our approach complements an unsupervised classifier, which exploits information about the cohesive structure of the discourse, with a supervised classifier. The latter can make use of a range of features and therefore base its classification decision on additional properties of the discourse, besides lexical cohesion. We showed that such a combined classifier can lead to a significant reduction of classification errors. Its performance can be improved further by iteratively increasing the training set in a bootstrapping loop and by adding additional examples of the literal class, which is typically the minority class. We found that such examples can be obtained automatically by extracting non-canonical variants of the target idioms from an unlabelled corpus.

Future work should look at improving the supervised classifier, which so far has an accuracy of 90%. While this is already pretty good, a more sophisticated model might lead to further improvements. For example, one could experiment with linguistically more informed features. While our initial studies in this direction were negative, careful feature engineering might lead to better results.

Acknowledgements

This work was funded by the Cluster of Excellence "Multimodal Computing and Interaction".

[16] Again, we disregard the extra literal examples here.

References

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*.

Colin Bannard. 2007. A measure of syntactic flexibility for automatically identifying multiword expressions in corpora. In *Proceedings of the ACL-07 Workshop on A Broader Perspective on Multiword Expressions*.

Julia Birke and Anoop Sarkar. 2006. A clustering approach for the nearly unsupervised recognition of nonliteral language. In *Proceedings of EACL-06*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT-98*.

Rudi L. Cilibrasi and Paul M.B. Vitanyi. 2007. The Google similarity distance. *IEEE Trans. Knowledge and Data Engineering*, 19(3):370–383.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurements*, 20:37–46.

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2007. Pulling their weight: Exploiting syntactic forms for the automatic identification of idiomatic expressions in context. In *Proceedings of the ACL-07 Workshop on A Broader Perspective on Multiword Expressions*.

Afsaneh Fazly and Suzanne Stevenson. 2006. Automatically constructing a lexicon of verb phrase idiomatic combinations. In *Proceedings of EACL-06*.

Afsaneh Fazly, Paul Cook, and Suzanne Stevenson. 2009. Unsupervised type and token identification of idiomatic expressions. *Computational Linguistics*, 35(1):61–103.

M.A.K. Halliday and R. Hasan. 1976. *Cohesion in English*. Longman House, New York.

Nathalie Japkowicz and Shaju Stephen. 2002. The class imbalance problem: A systematic study. *Intelligent Data Analysis Journal*, 6(5):429–450.

Graham Katz and Eugenie Giesbrecht. 2006. Automatic identification of non-compositional multiword expressions using latent semantic analysis. In *Proceedings of the ACL/COLING-06 Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*.

Anh-Cuong Le, Akira Shimazu, and Le-Minh Nguyen. 2006. Investigating problems of semi-supervised learning for word sense disambiguation. In *Proc. ICCPOL-06*.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *Proceedings of ACL-99*, pages 317–324.

Vincent Ng and Claire Cardie. 2003. Weakly supervised natural language learning without redundant views. In *Proc. of HLT-NAACL-03*.

Susanne Riehemann. 2001. *A Constructional Approach to Idioms and Word Formation*. Ph.D. thesis, Stanford University.

Caroline Sporleder and Linlin Li. 2009. Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of EACL-09*.

Aline Villavicencio, Valia Kordoni, Yi Zhang, Marco Idiart, and Carlos Ramisch. 2007. Validation and evaluation of automatically acquired multiword expressions for grammar engineering. In *Proceedings of EMNLP-07*.

Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing

Brian Roark[†] **Asaf Bachrach**[‡] **Carlos Cardenas**[◦] **and Christophe Pallier**[‡]
[†]Center for Spoken Language Understanding, Oregon Health & Science University
[‡] INSERM-CEA Cognitive Neuroimaging Unit, Gif sur Yvette, France [◦]MIT
roark@cslu.ogi.edu asafbac@gmail.com cardenas@mit.edu christophe@pallier.org

Abstract

A number of recent publications have made use of the incremental output of stochastic parsers to derive measures of high utility for psycholinguistic modeling, following the work of Hale (2001; 2003; 2006). In this paper, we present novel methods for calculating separate lexical and syntactic surprisal measures from a single incremental parser using a lexicalized PCFG. We also present an approximation to entropy measures that would otherwise be intractable to calculate for a grammar of that size. Empirical results demonstrate the utility of our methods in predicting human reading times.

1 Introduction

Assessment of linguistic complexity has played an important role in psycholinguistics and neurolinguistics for a long time, from the use of mean length of utterance and related scores in child language development (Klee and Fitzgerald, 1985), to complexity scores related to reading difficulty in human sentence processing studies (Yngve, 1960; Frazier, 1985; Gibson, 1998). Operationally, such linguistic complexity scores are derived via deterministic manual (human) annotation and scoring algorithms of language samples. Natural language processing has been employed to automate the extraction of such measures (Sagae et al., 2005; Roark et al., 2007), which can have high utility in terms of reduction of time required to annotate and score samples. More interestingly, however, novel data driven methods are being increasingly employed in this sphere, yielding language sample characterizations that *require* NLP in their derivation. For example, scores derived from variously estimated language models have been used to evaluate and classify language samples associated with neurodevelopmental or neurodegenerative disorders (Roark et al., 2007; Solorio and Liu, 2008; Gabani et al., 2009), as well as within general studies of human sentence processing (Hale, 2001; 2003; 2006). These scores cannot feasibly be derived by hand, but rather rely on large-scale statistical models and structured inference algorithms to be derived. This is quickly becoming an important application of NLP, making possible new methods in the study of human language processing in both typical and impaired populations.

The use of broad-coverage parsing for psycholinguistic modeling has become very popular recently. Hale (2001) suggested a measure (surprisal) derived from an Earley (1970) parser using a probabilistic context-free grammar (PCFG) for psycholinguistic modeling; and in later work (Hale, 2003; 2006) he suggested an alternate parser-derived measure (entropy reduction) that may also account for some human sentence processing performance. Recent work continues to advocate surprisal in particular as a very useful measure for predicting processing difficulty (Boston et al., 2008a; Boston et al., 2008b; Demberg and Keller, 2008; Levy, 2008), and the measure has been derived using a variety of incremental (left-to-right) parsing strategies, including an Earley parser (Boston et al., 2008a), the Roark (2001) incremental top-down parser (Demberg and Keller, 2008), and an n-best version of the Nivre et al. (2007) incremental dependency parser (Boston et al., 2008a; 2008b). Deriving such measures by hand, even for a relatively limited set of stimuli, is not feasible, hence parsing plays a critical role in this developing psycholinguistic enterprise.

There is no single measure that can account for all of the factors influencing human sentence processing performance, and some of the most recent work on using parser-derived measures for psycholinguistic modeling has looked to try to derive multiple, complementary measures. One of

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 324–333,
Singapore, 6-7 August 2009.

the key distinctions being looked at is syntactic versus lexical expectations (Gibson, 2006). For example, in Demberg and Keller (2008), trials were run deriving surprisal from the Roark (2001) parser under two different conditions: fully lexicalized parsing, and fully unlexicalized parsing (to pre-terminal part-of-speech tags). Boston et al. (2008a) capture a similar distinction by making use of an unlexicalized PCFG within an Earley parser and a fully lexicalized unlabeled dependency parser (Nivre et al., 2007). As Demberg and Keller (2008) point out, fully unlexicalized grammars ignore important lexico-syntactic information when deriving the "syntactic" expectations, such as subcategorization preferences of particular verbs, which are generally accepted to impact syntactic expectations in human sentence processing (Garnsey et al., 1997). Demberg and Keller argue, based on their results, for unlexicalized surprisal instead of lexicalized surprisal. Here we present a novel method for deriving separate syntactic and lexical surprisal measures from a fully lexicalized incremental parser, to allow for rich probabilistic grammars to be used to derive either measure, and demonstrate the utility of this method versus that of Demberg and Keller in empirical trials.

The use of large-scale lexicalized grammars presents a problem for using an Earley parser to derive surprisal or for the calculation of entropy as Hale (2003; 2006) defines it, because both methods require matrix inversion of a matrix with dimensionality the size of the non-terminal set. With very large lexicalized PCFGs, the size of the non-terminal set is too large for tractable matrix inversion. The use of an incremental, beam-search parser provides a tractable approximation to both measures. Incremental top-down and left-corner parsers have been shown to effectively (and efficiently) make use of non-local features from the left-context to yield very high accuracy syntactic parses (Roark, 2001; Henderson, 2003; Collins and Roark, 2004), and we will use such rich models to derive our scores.

In addition to teasing apart syntactic and lexical surprisal (defined explicitly in §3), we present an approximation to the full entropy that Hale (2003; 2006) used to define the entropy reduction hypothesis. Such an entropy measure is derived via a predictive step, advancing the parses independently of the input, as described in §3.3. We also present syntactic and lexical alternatives for this measure, and demonstrate the utility of making such a distinction for entropy as well as surprisal.

The purpose of this paper is threefold. First, to present a careful and well-motivated decomposition of lexical and syntactic expectation-based measures from a given lexicalized PCFG. Second, to explicitly document methods for calculating these and other measures from a specific incremental parser. And finally, to present some empirical validation of the novel measures from real reading time trials. We modified the Roark (2001) parser to calculate the discussed measures[1], and the empirical results in §4 show several things, including: 1) using a fully lexicalized parser to calculate syntactic surprisal and entropy provides higher predictive utility for reading times than these measures calculated via unlexicalized parsing (as in Demberg and Keller); and 2) syntactic entropy is a useful predictor of reading time.

2 Notation and preliminaries

A probabilistic context-free grammar (PCFG) $G = (V, T, S^\dagger, P, \rho)$ consists of a set of non-terminal variables V; a set of terminal items (words) T; a special start non-terminal $S^\dagger \in V$; a set of rule productions P of the form $A \rightarrow \alpha$ for $A \in V$, $\alpha \in (V \cup T)^*$; and a function ρ that assigns probabilities to each rule in P such that for any given non-terminal symbol $X \in V$, $\sum_\alpha \rho(X \rightarrow \alpha) = 1$.

For a given rule $A \rightarrow \alpha \in P$, let the function RHS return the right-hand side of the rule, i.e., $\text{RHS}(A \rightarrow \alpha) = \alpha$. Without loss of generality, we will assume that for every rule $A \rightarrow \alpha \in P$, one of two cases holds: either $\text{RHS}(A \rightarrow \alpha) \in T$ or $\text{RHS}(A \rightarrow \alpha) \in V^*$. That is, the right-hand side sequences consist of either (1) exactly one terminal item, or (2) zero or more non-terminals.

Let $W \in T^n$ be a terminal string of length n, i.e., $W = W_1 \ldots W_n$ and $|W| = n$. Let $W[i, j]$ denote the substring beginning at word W_i and ending at word W_j of the string. Then $W_{|W|}$ is the last word in the string, and $W[1, |W|]$ is the string as a whole. Adjacent strings represent concatenation, i.e., $W[1, i]W[i{+}1, j] = W[1, j]$. Thus $W[1, i]w$ represents the string where $W_{i+1} = w$.

We can define a "derives" relation (denoted $\Rightarrow_G$ for a given PCFG G) as follows: $\beta A \gamma \Rightarrow_G \beta\alpha\gamma$ if and only if $A \rightarrow \alpha \in P$. A string $W \in T^*$ is in the language of a grammar G if and only if $S^\dagger \stackrel{+}{\Rightarrow}_G W$, i.e., a sequence of one or more derivation steps yields the string from the start

[1] The parser version will be made publicly available.

non-terminal. A *leftmost* derivation begins with $S^\dagger$ and each derivation step replaces the leftmost non-terminal A in the yield with some α such that $A \rightarrow \alpha \in P$. For a leftmost derivation $S^\dagger \stackrel{*}{\Rightarrow}_G \alpha$, where $\alpha \in (V \cup T)^*$, the sequence of derivation steps that yield α can be represented as a tree, with the start symbol $S^\dagger$ at the root, and the "yield" sequence α at the leaves of the tree. A *complete* tree has only terminal items in the yield, i.e., $\alpha \in T^*$; a *partial* tree has some non-terminal items in the yield. With a leftmost derivation, the yield $\alpha = \beta\gamma$ partitions into an initial sequence of terminals $\beta \in T^*$ followed by a sequence of non-terminals $\gamma \in V^*$. For a complete derivation, $\gamma = \epsilon$; for a partial derivation $\gamma \in V^+$, i.e., one or more non-terminals. Let $\mathcal{T}(G, W[1,i])$ be the set of complete trees with $W[1,i]$ as the yield of the tree, given PCFG G.

A leftmost derivation D consists of a sequence of $|D|$ steps. Let D_i represent the i^{th} step in the derivation D, and $D[i,j]$ represent the subsequence of steps in D beginning with D_i and ending with D_j. Note that $D_{|D|}$ is the last step in the derivation, and $D[1,|D|]$ is the derivation as a whole. Each step D_i in the derivation is a rule in G, i.e., $D_i \in P$ for all i. The probability of the derivation and the corresponding tree is:

$$\rho(D) \quad = \quad \prod_{i=1}^{m} \rho(D_i) \tag{1}$$

Let $\mathcal{D}(G, W[1,i])$ be the set of all possible leftmost derivations D (with respect to G) such that $\text{RHS}(D_{|D|}) = W_i$. These are the set of partial leftmost derivations whose last step used a production with terminal W_i on the right-hand side. The prefix probability of $W[1,i]$ with respect to G is

$$\text{PrefixProb}_G(W[1,i]) \quad = \sum_{D \in \mathcal{D}(G,W[1,i])} \rho(D) \tag{2}$$

From this prefix probability, we can calculate the conditional probability of each word $w \in T$ in the terminal vocabulary, given the preceding sequence $W[1,i]$ as follows:

$$\begin{aligned} P_G(w \mid W[1,i]) &= \frac{\text{PrefixProb}_G(W[1,i]w)}{\sum_{w' \in T} \text{PrefixProb}_G(W[1,i]w')} \\ &= \frac{\text{PrefixProb}_G(W[1,i]w)}{\text{PrefixProb}_G(W[1,i])} \end{aligned} \tag{3}$$

This, in fact, is precisely the conditional probability that is used for language modeling for such applications as speech recognition and machine translation, which was the motivation for various syntactic language modeling approaches (Jelinek and Lafferty, 1991; Stolcke, 1995; Chelba and Jelinek, 1998; Roark, 2001).

As with language modeling, it is important to model the end of the string as well, usually with an explicit end symbol, e.g., </s>. For a string $W[1,i]$, we can calculate its prefix probability as shown above. To calculate its complete probability, we must sum the probabilities over the set of complete trees $\mathcal{T}(G, W[1,i])$. In such a way, we can calculate the conditional probability of ending the string with </s> given $W[1,i]$ as follows:

$$P_G(\text{</s>} \mid W[1,i]) = \frac{\sum_{D \in \mathcal{T}(G,W[1,i])} \rho(D)}{\text{PrefixProb}_G(W[1,i])} \tag{4}$$

2.1 Incremental top-down parsing

In this section, we review relevant details of the Roark (2001) incremental top-down parser, as configured for use here. As presented in Roark (2004), the probabilities in the PCFG are smoothed so that the parser is guaranteed not to fail due to garden pathing, despite following a beam search strategy. Hence there is always a non-zero prefix probability as defined in Eq. 2.

The parser follows a top-down leftmost derivation strategy. The grammar is factored so that every production has either a single terminal item on the right-hand side or is of the form A $\rightarrow$ B A-B, where A,B $\in V$ and the factored A-B category can expand to any sequence of children categories of A that can follow B. This factorization of n-ary productions continues to nullary factored productions, i.e., the end of the original production A $\rightarrow$ $\text{B}_1 \ldots \text{B}_n$ is signaled with an empty production A-B_1-$\ldots$-$\text{B}_n \rightarrow \epsilon$.

The parser maintains a set of possible connected derivations, weighted via the PCFG. It uses a beam search, whereby the highest scoring derivations are worked on first, and derivations that fall outside of the beam are discarded. The reader is referred to Roark (2001; 2004) for specifics about the beam search.

The model conditions the probability of each production on features extracted from the partial tree, including non-local node labels such as parents, grandparents and siblings from the left-context, as well as c-commanding lexical items. Hence this is a lexicalized grammar, though the incremental nature precludes a general head-first strategy, rather one that looks to the left-context for c-commanding lexical items.

To avoid some of the early prediction of structure, the version of the Roark parser that we used

performs an additional grammar transformation beyond the simple factorization already described – a selective left-corner transform of left-recursive productions (Johnson and Roark, 2000). In the transformed structure, slash categories are used to avoid predicting left-recursive structure until some explicit indication of modification is present, e.g., a preposition.

The final step in parsing, following the last word in the string, is to "complete" all non-terminals in the yield of the tree. All of these open non-terminals are composite factored categories, such as S-NP-VP, which are "completed" by rewriting to ϵ. The probability of these ϵ productions is what allows for the calculation of the conditional probability of ending the string, shown in Eq. 4.

One final note about the size of the non-terminal set and the intractability of exact inference for such a scenario. The non-terminal set not only includes the original atomic non-terminals of the grammar, but also any categories created by grammar factorization (S-NP) or the left-corner transform (NP/NP). Additionally, however, to remain context-free, the non-terminal set must include categories that incorporate non-local features used by the statistical model into their label, including parents, grandparents and sibling categories in the left-context, as well as c-commanding lexical heads. These non-local features must be made local by encoding them in the non-terminal labels, leading to a very large non-terminal set and intractable exact inference. Heavy smoothing is required when estimating the resulting PCFG. The benefit of such a non-terminal set is a rich model, which enables a more peaked statistical distribution around high quality syntactic structures and thus more effective pruning of the search space. The fully connected left-context produced by top-down derivation strategies provides very rich features for the stochastic parsing models. See Roark (2001; 2004) for discussion of these issues.

We now turn to measures that can be derived from the parser which may be of use for psycholinguistic modeling.

3 Parser and grammar derived measures

3.1 Surprisal

The *surprisal* at word W_i is the negative log probability of W_i given the preceding words. Using prefix probabilities, this can be calculated as:

$$S_G(W_i) \;=\; -\log \frac{\text{PrefixProb}_G(W[1,i])}{\text{PrefixProb}_G(W[1,i-1])} \quad (5)$$

Substituting equation 2 into this, we get

$$S_G(W_i) \;=\; -\log \frac{\sum_{D \in \mathcal{D}(G,W[1,i])} \rho(D)}{\sum_{D \in \mathcal{D}(G,W[1,i-1])} \rho(D)} \quad (6)$$

If we are using a beam-search parser, some of the derivations are pruned away. Let $\mathcal{B}(G, W[1,i]) \subseteq \mathcal{D}(G, W[1,i])$ be the set of derivations in the beam. Then the surprisal can be approximated as

$$S_G(W_i) \;\approx\; -\log \frac{\sum_{D \in \mathcal{B}(G,W[1,i])} \rho(D)}{\sum_{D \in \mathcal{B}(G,W[1,i-1])} \rho(D)} \quad (7)$$

Any pruning in the beam search will result in a *deficient* probability distribution, i.e., a distribution that sums to less than 1. Roark's thesis (2001) showed that the amount of probability mass lost for this particular approach is very low, hence this provides a very tight bound on the actual surprisal given the model.

3.2 Lexical and Syntactic surprisal

High surprisal scores result when the prefix probability at word W_i is low relative to the prefix probability at word W_{i-1}. Sometimes this is due to the identity of W_i, i.e., it is a surprising word given the context. Other times, it may not be the lexical identity of the word so much as the syntactic structure that must be created to integrate the word into the derivations. One would like to tease surprisal apart into "syntactic surprisal" versus "lexical surprisal", which would capture this intuition of the lexical versus syntactic dimensions to the score. Our solution to this has the beneficial property of producing two scores whose sum equals the original surprisal score.

The original surprisal score is calculated via sets of partial derivations at the point when each word W_i is integrated into the syntactic structure, $\mathcal{D}(G, W[1,i])$. We then calculate the ratio from point to point in sequence. To tease apart the lexical and syntactic surprisal, we will consider sets of partial derivations *immediately before* each word W_i is integrated into the syntactic structure, i.e., $D[1, |D|-1]$ for $D \in \mathcal{D}(G, W[1,i])$. Recall that the last derivation move for every derivation in the set is from the POS-tag to the lexical item. Hence the sequence of derivation moves that excludes the last one includes all structure except the word W_i. Then the syntactic surprisal is calculated as:

$$\text{Syn}S_G(W_i) = -\log \frac{\sum_{D \in \mathcal{D}(G,W[1,i])} \rho(D[1,|D|-1])}{\sum_{D \in \mathcal{D}(G,W[1,i-1])} \rho(D)} \quad (8)$$

and the lexical surprisal is calculated as:

$$\text{Lex}S_G(W_i) = -\log \frac{\sum_{D \in \mathcal{D}(G,W[1,i])} \rho(D)}{\sum_{D \in \mathcal{D}(G,W[1,i])} \rho(D[1,|D|-1])} \quad (9)$$

Note that the numerator of $\text{Syn}S_G(W_i)$ is the denominator of $\text{Lex}S_G(W_i)$, hence they sum to form total surprisal $S_G(W_i)$. As with total surprisal, these measures can be defined either for the full set $\mathcal{D}(G, W[1,i])$ or for a pruned beam of derivations $\mathcal{B}(G, W[1,i]) \subseteq \mathcal{D}(G, W[1,i])$.

Finally, we replicated the Demberg and Keller (2008) "unlexicalized" surprisal by replacing every lexical item in the training corpus with its POS-tag, and then parsing the POS-tags of the language samples rather than the words. This differs from our syntactic surprisal by having no lexical conditioning events for rule probabilities, and by having no ambiguity about the POS-tag of the lexical items in the string. We will refer to the resulting surprisal measure as "POS surprisal" to distinguish it from our syntactic surprisal measure.

3.3 Entropy

Entropy scores of the sort advocated by Hale (2003; 2006) involve calculation over the set of complete derivations consistent with the set of partial derivations. Hale performs this calculation efficiently via matrix inversion, which explains the use of relatively small-scale grammars with tractably sized non-terminal sets. Such methods are not tractable for the kinds of richly conditioned, large-scale PCFGs that we advocate using here. At each word in the string, the Roark (2001) top-down parser provides access to the weighted set of partial analyses in the beam; the set of complete derivations consistent with these is not immediately accessible, hence additional work is required to calculate such measures.

Let $H(\mathcal{D})$ be the entropy over a set of derivations $\mathcal{D}$, calculated as follows:

$$H(\mathcal{D}) = -\sum_{D \in \mathcal{D}} \frac{\rho(D)}{\sum_{D' \in \mathcal{D}} \rho(D')} \log \frac{\rho(D)}{\sum_{D' \in \mathcal{D}} \rho(D')} \quad (10)$$

If the set of derivations $\mathcal{D} = \mathcal{D}(G, W[1,i])$ is a set of partial derivations for string $W[1,i]$, then $H(\mathcal{D})$ is a measure of uncertainty over the partial derivations, i.e., the uncertainty regarding the correct analysis of what has already been processed. This can be calculated directly from the existing parser operations. If the set of derivations are the complete derivations *consistent* with the set of partial derivations – complete derivations that could occur over the set of possible continuations of the string – then this is a measure of the uncertainty about what is yet to come. We would like measures that can capture this distinction between (a) uncertainty of what has already been processed ("current ambiguity") versus (b) uncertainty of what is yet to be processed ("predictive entropy"). In addition, as with surprisal, we would like to tease apart the syntactic uncertainty versus lexical uncertainty.

To calculate the predictive entropy after word sequence $W[1,i]$, we modify the parser as follows: the parser extends the set of partial derivations to include all possible next words (the entire vocabulary plus </s>), and calculates the entropy over that set. This measure is calculated from just one additional word beyond the current word, and hence is an approximation to Hale's conditional entropy of grammatical continuations, which is over complete derivations. We will denote this as $H^1_G(W[1,i])$ and calculate it as follows:

$$H^1_G(W[1,i]) = H(\bigcup_{w \in T \cup \{\text{</s>}\}} \mathcal{D}(G, W[1,i]w)) \quad (11)$$

This is performing a predictive step that the baseline parser does not perform, extending the parses to all possible next words.

Unlike surprisal, entropy does not decompose straightforwardly into syntactic and lexical components that sum to the original composite measure. To tease apart entropy due to syntactic uncertainty versus that due to lexical uncertainty, we can define the set of derivations up to the preterminal (POS-tag) non-terminals as follows. Let $S(\mathcal{D}) = \{D[1,|D|-1] : D \in \mathcal{D}\}$, i.e., the set of derivations achieved by removing the last step of all derivations in $\mathcal{D}$. Then we can calculate a "syntactic" H^1_G as follows:

$$\text{Syn}H^1_G(W[1,i]) = H(\bigcup_{w \in T \cup \{\text{</s>}\}} S(\mathcal{D}(G, W[1,i]w))) \quad (12)$$

Finally, "lexical" H^1_G is defined in terms of the conditional probabilities derived from prefix probabilities as defined in Eq. 3.

$$\text{Lex}H^1_G(W[1,i]) = -\sum_{w \in T \cup \{\text{</s>}\}} P_G(w \mid W[1,i]) \log P_G(w \mid W[1,i]) \quad (13)$$

As a practical matter, these values are calculated within the Roark parser as follows. A "dummy" word is created that can be assigned every POS-tag, and the parser extends from the current state to this dummy word. (The beam threshold is greatly

expanded to allow for many possible extensions.) Then every word in the vocabulary is substituted for the word, and the appropriate probabilities calculated over the beam. Finally, the actual next word is substituted, the beam threshold is reduced to the actual working threshold, and the requisite number of analyses are advanced to continue parsing the string. This represents a significant amount of additional work for the parser – particularly for vocabulary sizes that we currently use, on the order of tens of thousands of words.

As with surprisal, we can calculate an "unlexicalized" version of the measure by training and parsing just to POS-tags. We will refer to this sort of entropy as "POS entropy".

4 Empirical validation

4.1 Subjects and stimuli

In order to test the psycholinguistic relevance of the different measures produced by the parser, we conducted a word by word reading experiment. 23 native speakers of English read 4 short texts (mean length: 883.5 words, 49.25 sentences). The texts were the written versions of narratives used in a parallel fMRI experiment making use of the same parser derived measures and whose results will be published in a different paper (Bachrach et al., 2009). The narratives contained a high density of syntactically complex structures (in the form of sentential embeddings, relative clauses and other non-local dependencies) but were constructed so as to appear highly natural. The modified version of the Roark parser, trained on the Brown Corpus section of the Penn Treebank (Marcus et al., 1993), was used to parse the different narratives and produce the word by word measures.

4.2 Procedure

Each narrative was presented line by line (certain sentences required more than one line) on a computer screen (Dell Optiplex 755 running Windows XP Professional) using Linger 2.88[2]. Each line contained 11.5 words on average. Each word would appear in its relative position on the screen. The subject would then be required to push a keyboard button to advance to the next word. The original word would then disappear and the following word appear in the subsequent position on the screen. After certain sentences a comprehension question would appear on the screen (10 per narrative). This was done in order to encourage subjects to pay attention and to provide data for a post-hoc evaluation of comprehension. After each narrative, subjects were instructed to take a short break (2 minutes on average).

4.3 Data analysis

The log (base 10) of the reaction times were analyzed using a linear mixed effects regression analysis implemented in the language R (Bates et al., 2008). Reaction times longer than 1500 ms and shorter than 150 ms (raw) were excluded from the analysis (4.8% of total data). Since button press latencies inferior to 150 ms must have been planned prior to the presentation of the word, we considered that they could not reflect stimulus driven effects. Data from the first and last words on each line were discarded.

The combined data from the 4 narratives was first modeled using a model which included order of word in the narrative[3], word length, parser-derived lexical surprisal, unigram frequency, bigram probability, syntactic surprisal, lexical entropy, syntactic entropy and mean number of parser derivation steps as numeric regressors. We also included the unlexicalized POS variants of syntactic surprisal and entropy, along the lines of Demberg and Keller (2008), as detailed in § 3. Table 1 presents the correlations between these mean-centered measures.

In addition, we modeled word class (open/closed) as a categorical factor in order to assess interaction between class and the variables of interest, since such an interaction has been observed in the case of frequency (Bradley, 1983). Finally, the random effect part of the model included intercepts for subjects, words and sentences. We report significant effects at the threshold $p < .05$.

Given the presence of significant interactions between lexical class (open/closed) and a number of the variables of interests, we decided to split the data set into open and closed class words and model these separately (linear mixed effects with the same numeric variables as in the full model).

In order to evaluate the usefulness of splitting total surprisal into lexical and syntactic components we compared, using a likelihood ratio test, a model where lexical and syntactic surprisal are modeled as distinct regressors to a model where a single regressor equal to their sum (total surprisal)

[2] http://tedlab.mit.edu/∼dr/Linger/readme.html

[3] This is a regressor to control for the trend of subjects to read faster later in the narrative.

Predictor	SynH	LexH	SynS	LexS	Freq	Bgrm	PosS	PosH	Step	WLen
Syntactic Entropy (SynH)	1.00	-0.26	0.00	0.24	-0.24	0.20	0.02	0.55	-0.05	0.18
Lexical Entropy (LexH)	-0.26	1.00	0.01	-0.40	0.43	-0.38	-0.03	0.02	0.11	-0.29
Syntactic Surprisal (SynS)	0.00	0.01	1.00	-0.12	0.08	0.18	0.77	0.21	0.38	-0.03
Lexical Surprisal (LexS)	0.24	-0.40	-0.12	1.00	-0.81	0.87	-0.10	-0.20	-0.35	0.64
Unigram Frequency (Freq)	-0.24	0.43	0.08	-0.81	1.00	-0.69	0.02	0.18	0.31	-0.72
Bigram Probability (Bgrm)	0.20	-0.38	0.18	0.87	-0.69	1.00	0.11	-0.11	-0.16	0.56
POS Surprisal (PosS)	0.02	-0.03	0.77	-0.10	0.02	0.11	1.00	0.22	0.32	0.02
POS Entropy (PosH)	0.55	0.02	0.21	-0.20	0.18	-0.11	0.22	1.00	0.16	-0.11
Derivation steps (Step)	-0.05	0.11	0.38	-0.35	0.31	-0.16	0.32	0.16	1.00	-0.24
Word Length (WLen)	0.18	-0.29	-0.03	0.64	-0.72	0.56	0.02	-0.11	-0.24	1.00

Table 1: Correlations between (mean-centered) predictors. Note that unigram frequencies were represented as logs, other scores as negative logs, hence the sign of the correlations.

was included. If the larger model provides a significantly better fit than the smaller model, this provides evidence that distinguishing between lexical and syntactic contributions to surprisal is relevant. Since total entropy is not a sum of syntactic and lexical entropy, an analogous test would not be valid in that case.

4.4 Results

All subjects successfully answered the comprehension questions (92.8% correct responses, S.D.=5.1). In the full model, we observed significant main effects of word class as well as of lexical surprisal, bigram probability, unigram frequency, syntactic entropy, POS entropy and of order in the narrative. Syntactic surprisal, lexical entropy and number of steps had no significant effect. Word length also had no significant main effect but interacted significantly with word class (open/closed). Word class also interacted significantly with lexical surprisal, unigram frequency and syntactic surprisal.

The presence of these interactions led us to construct models restricted to open and closed class items respectively. The estimated parameters are reported in Table 2. Reading time for open class words showed significant effects of unigram frequency, syntactic surprisal, syntactic entropy, POS entropy and order within the narrative. The positive effect of length approached significance. Reading time for closed class words exhibited significant effects of lexical surprisal, bigram probability, syntactic entropy and order in the narrative. Length had a non-significant negative effect, thus explaining the interaction observed in the full model.

The models with separate lexical and syntactic surprisal performed better than models including combined surprisal. For open class words, the Akaike's information criterion (AIC) was -54810 for the combined model and -54819 for the independent model (likelihood ratio test comparing the two, nested, models: $\chi^2(1)=10.7, p<.001$). For closed class items, combined model's AIC was -61467 and full model's AIC was -61469 (likelihood ratio test: $\chi^2(1)=3.54, p=0.06$).

	Estimate	Std. Error	t-value
Open-class			
(Intercept)	$2.40\times10^{+00}$	2.39×10^{-02}	100.4*
Lexical Surprisal	-1.99×10^{-04}	7.28×10^{-04}	-0.3
Word Length	8.97×10^{-04}	4.62×10^{-04}	1.9
Bigram	4.18×10^{-04}	5.27×10^{-04}	0.8
Unigram Freq	-2.43×10^{-03}	1.20×10^{-03}	-2.0*
Derivation Steps	-1.17×10^{-03}	9.02×10^{-04}	-1.3
Syntactic Entropy	2.55×10^{-03}	6.19×10^{-04}	4.1*
Lexical Entropy	3.96×10^{-04}	6.68×10^{-04}	0.6
Syntactic Surprisal	3.28×10^{-03}	9.71×10^{-04}	3.4*
Order in narrative	-1.43×10^{-05}	4.34×10^{-06}	-3.3*
POS Surprisal	-6.84×10^{-04}	8.11×10^{-04}	-0.8
POS Entropy	1.47×10^{-03}	6.05×10^{-04}	2.4*
Closed-class			
(Intercept)	$2.42\times10^{+00}$	2.32×10^{-02}	104.3*
Lexical Surprisal	2.02×10^{-03}	7.84×10^{-04}	2.6*
Word Length	-1.87×10^{-03}	1.13×10^{-03}	-1.7
Bigram	1.19×10^{-03}	4.94×10^{-04}	2.4*
Unigram Freq	1.69×10^{-03}	2.67×10^{-03}	0.6
Derivation Steps	3.01×10^{-04}	5.09×10^{-04}	0.6
Syntactic Entropy	3.15×10^{-03}	5.05×10^{-04}	6.2*
Lexical Entropy	1.83×10^{-04}	8.63×10^{-04}	0.2
Syntactic Surprisal	3.00×10^{-04}	8.35×10^{-04}	0.4
Order in narrative	-1.33×10^{-05}	3.99×10^{-06}	-3.3*
POS Surprisal	-6.46×10^{-04}	6.81×10^{-04}	-0.9
POS Entropy	6.63×10^{-04}	5.04×10^{-04}	1.3

Table 2: Estimated effects from mixed effects models on open and closed items (stars denote significance at p<.05)

4.5 Discussion

Our results demonstrate the relevance of modeling psycholinguistic processes using an incremental probabilistic parser, and the utility of the novel measures presented here. Of particular interest are: the significant effects of our syntactic entropy measure; the independent contributions of lexical surprisal, bigram probability and unigram frequency; and the differences between the predictions of the lexicalized parsing model and the unlexicalized (POS) parsing model.

The effect of entropy, or uncertainty regarding

the upcoming input independent of the surprise of that input, has been observed in non-linguistic tasks (Hyman, 1953; Bestmann et al., 2008) but to our knowledge has not been quantified before in the context of sentence processing. The usefulness of computational modeling is particularly evident in the case of entropy given the absence of any subjective procedure for its evaluation[4]. The results argue in favor of a predictive parsing architecture (Van Berkum et al., 2005). The approach to entropy here differs from the one described in Hale (2006) in a couple of ways. First, as discussed above, the calculation procedure is different – we focus on extending the derivations with just one word, rather than to all possible complete derivations. Second, and most importantly, Hale emphasizes entropy reduction (or the gain in information, given an input, regarding the rest of the sentence) as the correlate of cognitive cost while here we are interested in the amount of entropy itself (and not the size of change).

Interestingly, we observed only an effect of syntactic entropy, not lexical entropy. Recent ERP work has demonstrated that subjects do form specific lexical predictions in the context of sentence processing (Van Berkum et al., 2005; DeLong et al., 2005) and so we suspect that the absence of lexical entropy effect might be partly due to sparse data. Lexical surprisal and entropy were calculated using the internal state of a parser trained on the relatively small Brown corpus. Lexical entropy showed no significant effect while lexical surprisal affected only closed class words. This pattern of results might be due to the sparseness of the relevant information in such a small corpus (e.g., verb/object preferences) and the relevance of extra-textual dimensions (world knowledge, contextual information) to lexical-specific prediction. Closed class words are both more frequent (and hence better sampled) and are less sensitive to world knowledge, yet are often determined by the grammatical context.

Demberg and Keller (2008) made use of the same parsing architecture used here to compute a syntactic surprisal measure, but used an unlexicalized parser (down to POS-tags rather than words) for this score. Their "lexicalized" surprisal is equivalent to our total surprisal (lexical surprisal + syntactic surprisal), while their POS surprisal is derived from a completely different model. In contrast, our approach achieves lexical and syntactic measures from the same model. In order to evaluate the difference between the two approaches we added unlexicalized POS surprisal calculated along the lines of that paper to our model, along with an unlexicalized POS entropy from the same model. We found no effect of unlexicalized POS surprisal[5] and a significant (but relatively small) effect of unlexicalized POS entropy. While syntactic surprisal was correlated with POS surprisal (see Table 1) and syntactic entropy correlated with POS entropy, the fact that our syntactic measures still had a significant effect suggests that lexical information contributes towards the formation of syntactic expectations.

While the effect of surprisal calculated by an incremental top down parser has been already demonstrated (Demberg and Keller, 2008), our results argue for a distinction between the effect of lexical surprisal and that of syntactic surprisal without requiring unlexicalized parsing of the sort that Demberg and Keller advocate. It is important to keep in mind that this distinction between types of prediction (and as a consequence, prediction error) is not equivalent to the one drawn in the traditional cognitive science modularity debate, which has focused on the source of these predictions. We found a positive effect of syntactic surprisal in the case of open class words. The absence of an effect for closed class words remains to be explained.

We quantified word specific surprisal using 3 sources: the parser's internal state (lexical surprisal); probability given the preceding word (negative log bigram probability); and the unigram frequency of the word in a large corpus[6]. As can be observed in Table 1, these three measures are highly correlated[7]. This is the consequence of the smoothing in the estimation procedure but also relates to a more general fact about language use: overall, more frequent words are also words more expected to appear in a specific context (Anderson and Schooler, 1991). Despite these strong correlations, the three measures produced independent

[4]The Cloze procedure (Taylor, 1953) is one way to derive probabilities that could be used to calculate entropy, though this procedure is usually conducted with lexical elicitation, which would make syntactic entropy calculations difficult.

[5]We also ran the model including unlexicalized POS surprisal *without* our syntactic surprisal or syntactic entropy, and in this condition the unlexicalized POS surprisal measure had a nearly significant effect ($t = 1.85$), which is consistent with the results in Boston et al. (2008a) and Demberg and Keller (2008).

[6]The unigram frequencies came from the HAL corpus (Lund and Burgess, 1996). All other statistical models were estimated from the Brown Corpus.

[7]Unigram frequencies were represented as logs, the others as negative logs, hence the sign of the correlations.

effects. Unigram frequency had a significant effect for open class words while bigram probability and lexical surprisal each had an effect on reading time of closed class items. Bigram probability has been often found to affect reading time using eye movement measures. This is the first study to demonstrate an additional effect of contextual surprisal given the preceding sentential context (lexical surprisal). Demberg and Keller found no effect for surprisal once bigram and unigram probabilities were included in the model but, importantly, they did not distinguish lexical and syntactic surprisal, rather "lexicalized" and "unlexicalized" surprisal.

5 Summary

We have presented novel methods for teasing apart syntactic and lexical surprisal from a fully lexicalized parser, as well as for extending the operation of a predictive parser to capture novel entropy measures that are also shown to be relevant to psycholinguistic modeling. Such automatic methods provide psycholinguistically relevant measures that are intractable to calculate by hand. The empirical validation presented here demonstrated that the new measures – particularly syntactic entropy and syntactic surprisal – have high utility for modeling human reading time data. Our approach to calculating syntactic surprisal, based on fully lexicalized parsing, provided significant effects, while the POS-tag based (unlexicalized) surprisal – of the sort used in Boston et al. (2008a) and Demberg and Keller (2008) – did not provide a significant effect in our trials. Further, we showed an effect of lexical surprisal for closed class words even when combined with unigram and bigram probabilities in the same model. This work contributes to the important, developing enterprise of leveraging data-driven NLP approaches to derive new measures of high utility for psycholinguistic and neuropsychological studies.

Acknowledgments

Thanks to Michael Collins, John Hale and Shravan Vasishth for valuable discussions about this work. This research was supported in part by NSF Grant #BCS-0826654. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

References

J.R. Anderson and L.J. Schooler. 1991. Reflections of the environment in memory. *Psychological Science*, 2(6):396–408.

A. Bachrach, B. Roark, A. Marantz, S. Whitfield-Gabrieli, C. Cardenas, and J.D.E. Gabrieli. 2009. Incremental prediction in naturalistic language processing: An fMRI study. *In preparation*.

D. Bates, M. Maechler, and B. Dai, 2008. *lme4: Linear mixed-effects models using S4 classes*. R package version 0.999375-20.

S. Bestmann, L.M. Harrison, F. Blankenburg, R.B. Mars, P. Haggard, and K.J. Friston. 2008. Influence of uncertainty and surprise on human corticospinal excitability during preparation for action. *Current Biology*, 18:775–780.

M. Ferrara Boston, J.T. Hale, R. Kliegl, U. Patil, and S. Vasishth. 2008a. Parsing costs as predictors of reading difficulty: An evaluation using the Potsdam sentence corpus. *Journal of Eye Movement Research*, 2(1):1–12.

M. Ferrara Boston, J.T. Hale, R. Kliegl, and S. Vasishth. 2008b. Surprising parser actions and reading difficulty. In *Proceedings of ACL-08:HLT, Short Papers*, pages 5–8.

D.C. Bradley. 1983. *Computational Distinctions of Vocabulary Type*. Indiana University Linguistics Club, Bloomington.

C. Chelba and F. Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proceedings of ACL-COLING*, pages 225–231.

M.J. Collins and B. Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, pages 111–118.

K.A. DeLong, T.P. Urbach, and M. Kutas. 2005. Probabilistic word pre-activation during language comprehension inferred from electrical brain activity. *Nature Neuroscience*, 8(8):1117–1121.

V. Demberg and F. Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.

J. Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 6(8):451–455.

L. Frazier. 1985. Syntactic complexity. In D.R. Dowty, L. Karttunen, and A.M. Zwicky, editors, *Natural Language Parsing*. Cambridge University Press, Cambridge, UK.

K. Gabani, M. Sherman, T. Solorio, and Y. Liu. 2009. A corpus-based approach for the prediction of language impairment in monolingual English and Spanish-English bilingual children. In *Proceedings of NAACL-HLT*.

S.M. Garnsey, N.J. Pearlmutter, E. Myers, and M.A. Lotocky. 1997. The contributions of verb bias and plausibility to the comprehension of temporarily ambiguous sentences. *Journal of Memory and Language*, 37(1):58–93.

E. Gibson. 1998. Linguistic complexity: locality of syntactic dependencies. *Cognition*, 68(1):1–76.

E. Gibson. 2006. The interaction of top-down and bottom-up statistics in the resolution of syntactic category ambiguity. *Journal of Memory and Language*, 54(3):363–388.

J.T. Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the 2nd meeting of NAACL.*

J.T. Hale. 2003. The information conveyed by words in sentences. *Journal of Psycholinguistic Research*, 32(2):101–123.

J.T. Hale. 2006. Uncertainty about the rest of the sentence. *Cognitive Science*, 30(4):643–672.

J. Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proceedings of HLT-NAACL*, pages 24–31.

R. Hyman. 1953. Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology: General*, 45(3):188–96.

F. Jelinek and J. Lafferty. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–323.

M. Johnson and B. Roark. 2000. Compact non-left-recursive grammars using the selective left-corner transform and factoring. In *Proceedings of COLING*, pages 355–361.

T. Klee and M.D. Fitzgerald. 1985. The relation between grammatical development and mean length of utterance in morphemes. *Journal of Child Language*, 12:251–269.

R. Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

K. Lund and C. Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28:203–208.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

B. Roark, M. Mitchell, and K. Hollingshead. 2007. Syntactic complexity measures for detecting mild cognitive impairment. In *Proceedings of BioNLP Workshop at ACL*, pages 1–8.

B. Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

B. Roark. 2004. Robust garden path parsing. *Natural Language Engineering*, 10(1):1–24.

K. Sagae, A. Lavie, and B. MacWhinney. 2005. Automatic measurement of syntactic development in child langugage. In *Proceedings of ACL*, pages 197–204.

T. Solorio and Y. Liu. 2008. Using language models to identify language impairment in Spanish-English bilingual children. In *Proceedings of BioNLP Workshop at ACL*, pages 116–117.

A. Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–202.

W.L. Taylor. 1953. Cloze procedure: A new tool for measuring readability. *Journalism Quarterly*, 30:415–433.

J.J.A. Van Berkum, C.M. Brown, P. Zwitserlood, V.Kooijman, and P. Hagoort. 2005. Anticipating upcoming words in discourse: Evidence from ERPs and reading times. *Learning and Memory*, 31(3):443–467.

V.H. Yngve. 1960. A model and an hypothesis for language structure. *Proceedings of the American Philosophical Society*, 104:444–466.

It's Not You, it's Me: Detecting Flirting and its Misperception in Speed-Dates

Rajesh Ranganath
Computer Science Department
Stanford University
rajeshr@cs.stanford.edu

Dan Jurafsky
Linguistics Department
Stanford University
jurafsky@stanford.edu

Dan McFarland
School of Education
Stanford University
dmcfarla@stanford.edu

Abstract

Automatically detecting human social intentions from spoken conversation is an important task for dialogue understanding. Since the social intentions of the speaker may differ from what is perceived by the hearer, systems that analyze human conversations need to be able to extract both the perceived and the intended social meaning. We investigate this difference between intention and perception by using a spoken corpus of speed-dates in which both the speaker and the listener rated the speaker on flirtatiousness. Our flirtation-detection system uses prosodic, dialogue, and lexical features to detect a speaker's intent to flirt with up to 71.5% accuracy, significantly outperforming the baseline, but also outperforming the human interlocuters. Our system addresses lexical feature sparsity given the small amount of training data by using an autoencoder network to map sparse lexical feature vectors into 30 compressed features. Our analysis shows that humans are very poor perceivers of intended flirtatiousness, instead often projecting their own intended behavior onto their interlocutors.

1 Introduction

Detecting human social meaning is a difficult task for automatic conversational understanding systems. One cause of this difficulty is the pervasive difference between intended social signals and the uptake by the perceiver. The cues that a speaker may use to attempt to signal a particular social meaning may not be the cues that the hearer focuses on, leading to misperception.

In order to understand the impact of this difference between perception and intention, in this paper we describe machine learning models that can detect both the social meaning intended by the speaker and the social meaning perceived by the hearer. Automated systems that detect and model these differences can lead both to richer socially aware systems for conversational understanding and more sophisticated analyses of conversational interactions like meetings and interviews.

This task thus extends the wide literature on social meaning and its detection, including the detection of emotions such as annoyance, anger, sadness, or boredom (Ang et al., 2002; Lee and Narayanan, 2002; Liscombe et al., 2003), speaker characteristics such as charisma (Rosenberg and Hirschberg, 2005), personality features like extroversion or agreeability (Mairesse et al., 2007; Mairesse and Walker, 2008), speaker depression or stress (Rude et al., 2004; Pennebaker and Lay, 2002; Cohn et al., 2004), and dating willingness or liking (Madan et al., 2005; Pentland, 2005).

We chose to work on the domain of *flirtation* in speed-dating. Our earlier work on this corpus showed that it is possible to detect whether speakers are perceived as flirtatious, awkward, or friendly with reasonable accuracy (Jurafsky et al., 2009). In this paper we extend that work to detect whether speakers themselves intended to flirt, explore the differences in these variables, and explore the ability and inability of humans to correctly perceive the flirtation cues.

While many of the features that we use to build these detectors are drawn from the previous literature, we also explore new features. Conventional methods for lexical feature extraction, for example, generally consist of hand coded classes of words related to concepts like *sex* or *eating* (Pennebaker et al., 2007). The classes tend to perform well in their specific domains, but may not be robust across domains, suggesting the need for unsupervised domain-specific lexical feature extraction. The naive answer to extracting domain-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 334–342,
Singapore, 6-7 August 2009.

specific lexical features would just be to throw counts for every word into a huge feature vector, but the curse of dimensionality rules this method out in small training set situations. We propose a new solution to this problem, using an unsupervised deep autoencoder to automatically compress and extract complex high level lexical features.

2 Dataset

Our experiments make use of the SpeedDate Corpus collected by the third author, and described in Jurafsky et al. (2009). The corpus is based on three speed-dating sessions run at an American university in 2005, inspired by prior speed-dating research (Madan et al., 2005). The graduate student participants volunteered to be in the study and were promised emails of persons with whom they reported mutual liking. All participants wore audio recorders on a shoulder sash, thus resulting in two audio recordings of the approximately 1100 4-minute dates. Each date was conducted in an open setting where there was substantial background noise. This noisy audio was thus hand-transcribed and turn start and end were hand-aligned with the audio. In addition to the audio, the corpus includes various attitude and demographic questions answered by the participants.

Each speaker was also asked to report how often their date's speech reflected different conversational styles (awkward, flirtatious, funny, assertive) on a scale of 1-10 (1=never, 10=constantly): "How often did the other person behave in the following ways on this 'date'?". In addition they were also asked to rate their own intentions: "How often did you behave in the following ways on this 'date'?" on a scale of 1-10.

In this study, we focus on the *flirtation* ratings, examining how often each participant said they were flirting, as well as how often each participant was judged by the interlocutor as flirting.

Of the original 1100 dates only 991 total dates are in the SpeedDate corpus due to various losses during recording or processing. The current study focuses on 946 of these, for which we have complete audio, transcript, and survey information.

3 Experiment

To understand how the perception of flirting differs from the intention of flirting, we trained binary classifiers to predict both perception and intention. In each date, the speaker and the interlocutor both labeled the speaker's behavioral traits on a Likert scale from 1-10. To generate binary responses we took the top ten percent of Likert ratings in each task and labeled those as positive examples. We similarly took the bottom ten percent of Likert ratings and labeled those as negative examples. We ran our binary classification experiments to predict this output variable. Our experiments were split by gender. For the female experiment the speaker was female and the interlocutor was male, while for the male experiment the speaker was male and the interlocutor was female.

For each speaker side of each 4-minute conversation, we extracted features from wavefiles and transcripts, as described in the next section. We then trained four separate binary classifiers (for each gender for both perception and intention).

4 Feature Descriptions

We used the features reported by Jurafsky et al. (2009), which are briefly summarized here. The features for a conversation side thus indicate whether a speaker who talks a lot, laughs, is more disfluent, has higher F0, etc., is more or less likely to consider themselves flirtatious, or be considered flirtatious by the interlocutor. We also computed the same features for the *alter* interlocutor. *Alter* features thus indicate the conversational behavior of the speaker talking with an interlocutor they considered to be flirtatious or not.

4.1 Prosodic Features

F0 and RMS amplitude features were extracted using Praat scripts (Boersma and Weenink, 2005). Since the start and end of each turn were time-marked by hand, each feature was easily extracted over a turn, and then averages and standard deviations were taken over the turns in an entire conversation side. Thus the feature F0 MIN for a conversation side was computed by taking the F0 min of each turn in that side (not counting zero values of F0), and then averaging these values over all turns in the side. F0 MIN SD is the standard deviation across turns of this same measure.

4.2 Dialogue and Disfluency Features

A number of discourse features were extracted, following Jurafsky et al. (2009) and the dialogue literature. The dialog acts shown in Table 2 were detected by hand-built regular expressions, based on analyses of the dialogue acts in the

F0 MIN	minimum (non-zero) F0 per turn, averaged over turns
F0 MIN SD	standard deviation from F0 min
F0 MAX	maximum F0 per turn, averaged over turns
F0 MAX SD	standard deviation from F0 max
F0 MEAN	mean F0 per turn, averaged over turns
F0 MEAN SD	standard deviation (across turns) from F0 mean
F0 SD	standard deviation (within a turn) from F0 mean, averaged over turns
F0 SD SD	standard deviation from the f0 sd
PITCH RANGE	f0 max - f0 min per turn, averaged over turns
PITCH RANGE SD	standard deviation from mean pitch range
RMS MIN	minimum amplitude per turn, averaged over turns
RMS MIN SD	standard deviation from RMS min
RMS MAX	maximum amplitude per turn, averaged over turns
RMS MAX SD	standard deviation from RMS max
RMS MEAN	mean amplitude per turn, averaged over turns
RMS MEAN SD	standard deviation from RMS mean
TURN DUR	duration of turn in seconds, averaged over turns
TIME	total time for a speaker for a conversation side, in seconds
RATE OF SPEECH	number of words in turn divided by duration of turn in seconds, averaged over turns

Table 1: Prosodic features from Jurafsky et al. (2009) for each conversation side, extracted using Praat from the hand-segmented turns of each side.

hand-labeled Switchboard corpus of dialog acts. *Collaborative completions*, turns where a speaker completes the utterance begun by the alter, were detected by finding sentences for which the first word of the speaker was extremely predictable from the last two words of the previous speaker, based on a trigram grammar trained on the Treebank 3 Switchboard transcripts. Laughter, disfluencies, and overlap were all marked in the transcripts by the transcribers.

4.3 Lexical Features

We drew our lexical features from the LIWC lexicons of Pennebaker et al. (2007), the standard for social psychological analysis of lexical features. We chose ten LIWC categories that have proven useful in detecting personality-related features (Mairesse et al., 2007): *Anger, Assent, Ingest, Insight, Negemotion, Sexual, Swear, I, We,* and *You*. We also added two new lexical features: "past tense auxiliary", a heuristic for automatically detecting narrative or story-telling behavior, and *Metadate*, for discussion about the speed-date itself. The features are summarized in Table 3.

4.4 Inducing New Lexical Features

In Jurafsky et al. (2009) we found the LIWC lexical features less useful in detecting social meaning than the dialogue and prosodic features, perhaps because lexical cues to flirtation lie in different classes of words than previously investigated. We therefore investigated the induction of lexical features from the speed-date corpus, using a probabilisitic graphical model.

We began with a pilot investigation to see whether lexical cues were likely to be useful; with a small corpus, it is possible that lexical features are simply too sparse to play a role given the limited data. The pilot was based on using Naive Bayes with word existence features (binomial Naive Bayes). Naive Bayes assumes all features are conditionally independent given the class, and is known to perform well with small amounts of data (Rish, 2001). Our Naive Bayes pilot system performed above chance, suggesting that lexical cues are indeed informative.

A simple approach to including lexical features in our more general classification system would be to include the word counts in a high dimensional feature vector with our other features. This method, unfortunately, would suffer from the well-known high dimensionality/small training set problem. We propose a method for building a much smaller number of features that would nonetheless capture lexical information. Our approach is based on using autoencoders to construct high level lower dimension features from the words in a nonlinear manner.

A deep autoencoder is a hierarchichal graphical model with multiple layers. Each layer consists of a number of units. The input layer has the same number of units as the output layer, where the output layer is the model's reconstruction of the input layer. The number of units in the intermediate layers tends to get progressively smaller to produce a compact representation.

We defined our autoencoder with visible units modeling the probabilities of the 1000 most common words in the conversation for the speaker and the probabilities of the 1000 most common words for the interlocutor (after first removing a stop list of the most common words). We train a deep autoencoder with stochastic nonlinear feature detectors and linear feature detectors in the final layer. As shown in Figure 1, we used a 2000-1000-500-250-30 autoencoder. Autoen-

BACKCHANNELS	number of backchannel utterances in side (*Uh-huh.*, *Yeah.*, *Right.*, *Oh, okay.*)
APPRECIATIONS	number of appreciations in side (*Wow*, *That's true*, *Oh, great*)
QUESTIONS	number of questions in side
NTRI	repair question (Next Turn Repair Indicator) (*Wait*, *Excuse me*)
COMPLETION	(an approximation to) utterances that were 'collaborative completions'
LAUGH	number of instances of laughter in side
TURNS	total number of turns in side
DISPREFERRED	(approximation to) dispreferred responses, beginning with discourse marker *well*
UH/UM	total number of filled pauses (*uh* or *um*) in conversation side
RESTART	total number of disfluent restarts in conversation side
OVERLAP	number of turns in side which the two speakers overlapped

Table 2: Dialog act and disfluency features from Jurafsky et al. (2009).

TOTAL WORDS	total number of words
PAST TENSE	uses of past tense auxiliaries *was, were, had*
METADATE	*horn, date, bell, survey, speed, form, questionnaire, rushed, study, research*
YOU	*you, you'd, you'll, your, you're, yours, you've* (not counting *you know*)
WE	*lets, let's, our, ours, ourselves, us, we, we'd, we'll, we're, we've*
I	*I'd, I'll, I'm, I've, me, mine, my, myself* (not counting *I mean*)
ASSENT	*yeah, okay, cool, yes, awesome, absolutely, agree*
SWEAR	*hell, sucks, damn, crap, shit, screw, heck, fuck**
INSIGHT	*think*/thought, feel*/felt, find/found, understand*, figure*, idea*, imagine, wonder*
ANGER	*hate/hated, hell, ridiculous*, stupid, kill*, screwed, blame, sucks, mad, bother, shit*
NEGEMOTION	*bad, weird, hate, crazy, problem*, difficult, tough, awkward, boring, wrong, sad, worry,*
SEXUAL	*love*, passion*, virgin, sex, screw*
INGEST	*food, eat*, water, bar/bars, drink*, cook*, dinner, coffee, wine, beer, restaurant, lunch, dish*

Table 3: Lexical features from Jurafsky et al. (2009). Each feature value is a total count of the words in that class for each conversation side; asterisks indicate including suffixed forms (e.g., *love, loves, loving*). All except the first three are from LIWC (Pennebaker et al., 2007) (modified slightly, e.g., by removing *you know* and *I mean*). The last five classes include more words in addition to those shown.

coders tend to perform poorly if they are initialized incorrectly, so we use the Restricted Boltzmann Machine (RBM) pretraining procedure described in Hinton and Salakhutdinov (2006) to initialize the encoder. Each individual RBM is trained using contrastive divergence as an update rule which has been shown to produce reasonable results quickly (Hinton et al., 2006). Finally, we use backpropagation to fine tune the weights of our encoder by minimizing the cross entropy error. To extract features from each conversation, we sample the code layer (30 unit layer in our encoder) with the visible units corresponding to the most common word probabilities from that document, creating 30 new features that we can use for classification. The conditional distributions of the first layer features can be given by the softmax of the activations for each gender:

$$p(v_i|h) = \frac{\exp(bias_i + \sum_j h_j * w_{ij})}{\sum_{k \in K} \exp(bias_k + \sum_j v_j * w_{kj})} \quad (1)$$

$$p(h_j|v) = \frac{1}{1 + \exp(bias_(j) + \sum_i v_i * w_{ij})} \quad (2)$$

where K is the set of all the units representing the same speaker as i [1], v_i is the ith visible unit, h_j is the jth hidden unit, w_{ij} is the weight between visible unit i and hidden unit j, and bias$_m$ is the offset of unit m. Intuitively, this means that the probability that a hidden unit is activated by the visible layer is sigmoid of the weighted sum of all the visible units plus the unit's bias term. Similarly, the visible units are activated through a weighted sum of the hidden units, but they undergo an additional normalization (softmax) over all the other visible units from the speaker to effectively model the multinomial distribution from each speaker. Since in a RBM hidden units are conditionally independent given the visible units, and visible units are

[1]The visible unit i models word probabilities of either the speaker or the interlocutor, so the softmax is done over the distribution of words for the speaker that unit i is modeling.

conditionally independent given hidden layer, the above equations completely specify the first layer of the model.

To account for the fact that each visible unit in the first layer contained 1000 observations from the underlying distribution we upweighted our features by that factor. During pretraining the "training data" for the higher layers is the activation probabilities of the hidden units of layer directly below when driven by that layer's input data. The intermediate layers in the model are symmetric where the activation probabilities for both the visible and hidden units are of the same form as $p(h_j|v)$ in layer 1. To produce real valued features in the code layer we used linear hidden units. In addition to the likelihood portion of the objective we penalized large weights by using l2 regularization and penalize all weights by applying a small constant weight cost that gets applied at every update. After training to find a good initial point for the autoencoder we unroll the weights and use backpropogation to fine tune our autoencoder.

While interpreting high level nonlinear features can be challenging, we did a pilot analysis of one of the 30 features fixing a large (positive or negative) weight on the feature unit (code layer) and sampling the output units.

The top weighted words for a positive weight are: *O_did, O_live, S_did, S_friends, S_went, O_live, S_lot, S_wait, O_two*, and *O_wasn't* (S for speaker and O for interlocutor). The top weighted words for a negative weight are: *S_long, O_school, S_school, S_phd, O_years, S_years, O_stanford, S_lot, O_research, O_interesting* and *O_education*. At least for this one feature, a large positive value seemed to indicate the prevalence of questions (*wait, did*) or storytelling (
em live, wasn't). A large negative weight indicates the conversation focused on the mundane details of grad student life.

5 Classification

Before performing the classification task, we preprocessed the data in two ways. First, we standardized all the variables to have zero mean and unit variance. We did this to avoid imposing a prior on any of the features based on their numerical values. Consider a feature A with mean 100 and a feature B with mean .1 where A and B are correlated with the output. Since the SVM problem minimizes the norm of the weight vector, there

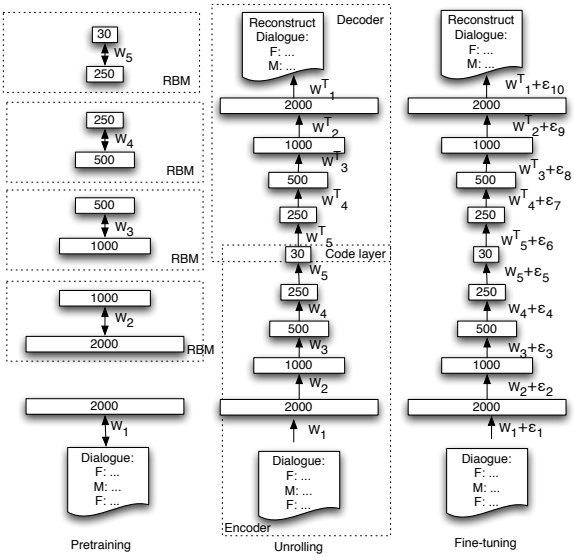


Figure 1: Pretraining is a fully unsupervised procedure that trains an RBM at each layer. Once the pretraining of one layer is complete, the top layer units are used as input to the next layer. We then fine-tune our weights using backprop. The 30 features are extracted from the code layer.

is a bias to put weight on feature A because intuitively the weight on feature B would need to be 1000 times larger to carry the same effect. This argument holds similarly for the reduction to unit variance. Second, we removed features correlated greater than .7. One goal of removing correlated features was to remove as much colinearity as possible from the regression so that the regression weights could be ranked for their importance in the classification. In addition, we hoped to improve classification because a large number of features require more training examples (Ng, 2004). For example for perception of female flirt we removed the number of turns by the alter (*O_turns*) and the number of sentence from the ego (*S_sentences*) because they were highly correlated with *S_turns*.

To ensure comparisons (see Section 7) between the interlocutors' ratings and our classifier (and because of our small dataset) we use k-fold cross validation to learn our model and evaluate our model. We train our binary model with the top ten percent of ratings labeled as positive class examples and bottom ten percent of ratings as the negative class examples. We used five-fold cross validation in which the data is split into five equal folds of size 40. We used four of the folds for training and one for test. K-fold cross validation does this in a round robin manner so every exam-

ple ends up in the test set. This yields a datasplit of 160 training examples and 40 test examples. To ensure that we were not learning something specific to our data split, we randomized our data ordering.

For classification we used a support vector machine (SVM). SVMs generally do not produce explicit feature weights for analysis because they are a kernelized classifier. We solved the linear C-SVM problem. Normally the problem is solved in the dual form, but to facilitate feature analysis we expand back to the primal form to retrieve w, the weight vector. Our goal in the C-SVM is to solve, in primal form,

$$\min_{\gamma,w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i$$
$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i,\ i = 1,\ldots,m$$
$$\xi_i \geq 0,\ i = 1,\ldots,m \quad (3)$$

where m is the number of training examples, $x^{(i)}$ is the ith training examples, and $y^{(i)}$ is the ith class (1 for the positive class, -1 for the negative class). The ξ_i are the slack variables that allow this algorithm to work for non linearly separable datasets.

A test example is classified by looking at the sign of $y(x) = w^T x^{(test)} + b$. To explore models that captured interactions, but do not allow for direct feature analysis we solved the C-SVM problem using a radial basis function (RBF) as a kernel (Scholkopf et al., 1997). Our RBF kernel is based on a Gaussian with unit variance.

$$K(x^{(i)}, x^{(j)}) = \exp(\frac{-||x^{(i)} - x^{(j)}||^2}{2\sigma}) \quad (4)$$

In this case predictions can be made by looking at $y(x^{(test)}) = \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} \text{rbf}(x^{(i)}, t^{(test)}) + b$, where each $\alpha^{(i)}$, for $i = 1,\ldots,m$ is a member of the set of dual variables that comes from transforming the primal form into the dual form. The SVM kernel trick allows us to explore higher dimensions while limiting the curse of dimensionality that plagues small datasets like ours.

We evaluated both our linear C-SVM and our radial basis function C-SVM using parameters learned on the training sets by computing the accuracy on the test set. Accuracy is the number of correct examples / total number of test examples. We found that the RBM classifier that handled interaction terms outperformed linear methods like logistic regression.

For feature weight extraction we aggregated the feature weights calculated from each of the test folds by taking the mean between them.[2]

6 Results

We report in Table 4 the results for detecting flirt intention (whether a speaker said they were flirting) as well as flirt perception (whether the listener said the speaker was flirting).

	Flirt Intention		Flirt Perception	
	by M	by F	of M	of F
RBM SVM	61.5%	70.0%	77.0%	59.5%
+autoencoder features	69.0%	71.5%	79.5%	68.0%

Table 4: Accuracy of binary classification of each conversation side, where chance is 50%. The first row uses all the Jurafsky et al. (2009) features for both the speaker and interlocutor. The second row adds the new autoencoder features.

In our earlier study of flirt perception, we achieved 71% accuracy for men and 60% for women (Jurafsky et al., 2009). Our current numbers for flirt perception are much better for both men (79.5%), and women (68.0%). The improvement is due both to the new autoencoder features and the RBF kernel that considers feature interactions (feature interactions were not included in the logistic regression classifiers of Jurafsky et al. (2009)).

Our number for flirt intention are 69.0% for men and 71.5% for women. Note that our accuracies are better for detecting women's intentions as well as women's perceptions (of men) than men's intentions and perceptions.

7 Feature Analysis

We first considered the features that helped classification of flirt intention. Table 5 shows feature weights for the features (features were normed so weights are comparable), and is summarized in the following paragraphs:

• Men who say they are flirting ask more questions, and use more *you* and *we*. They laugh more, and use more sexual, anger, and negative emotional words. Prosodically they speak faster, with higher pitch, but quieter (lower intensity min).

[2]We could not use the zero median criteria used in Jurafsky et al. (2009) because C-SVMs under the l-2 metric provide no sparse weight guarantees.

FEMALE FLIRT		MALE FLIRT	
O_backchannel	-0.0369	S_you	0.0279
S_appreciation	-0.0327	S_negemotion	0.0249
O_appreciation	-0.0281	S_we	0.0236
O_question	0.0265	S_anger	0.0190
O_avimin	-0.0249	S_sexual	0.0184
S_turns	-0.0247	O_negemotion	0.0180
S_backchannel	-0.0245	O_avpmax	0.0174
O_you	0.0239	O_swear	0.0172
S_avtndur	0.0229	O_laugh	0.0164
S_avpmin	-0.0227	O_wordcount	0.0151
O_rate	0.0212	S_laugh	0.0144
S_laugh	0.0204	S_rate	0.0143
S_wordcount	0.0192	S_well	0.0131
S_well	0.0192	S_question	0.0131
O_negemotion	0.019	O_sexual	0.0128
S_repair_q	0.0188	S_completion	0.0128
O_sexual	0.0176	S_avpmax	0.011
O_overlap	-0.0176	O_completion	0.010
O_sdpmean	0.0171	O_sdimin	0.010
O_avimax	-0.0151	O_metatalk	-0.012
S_avpmean	-0.015	S_sdpsd	-0.015
S_question	-0.0146	S_avimin	-0.015
O_sdimin	0.0136	S_backchannel	-0.022
S_avpmax	0.0131		
S_we	-0.013		
S_I	0.0117		
S_assent	0.0114		
S_metatalk	-0.0107		
S_sexual	0.0105		
S_avimin	-0.0104		
O_uh	-0.0102		

Table 5: Feature weights (mean weights of the randomized runs) for the predictors with $|weight| > 0.01$ for the male and female classifiers. An S prefix indicates features of the speaker (the candidate flirter) while an O prefix indicates features of the other. Weights for autoencoder features were also significant but are omitted for compactness.

Features of the alter (the woman) that helped our system detect men who say they are flirting include the woman's laughing, sexual words or swear words, talking more, and having a higher f0 (max).

• Women who say they are flirting have a much expanded pitch range (lower pitch min, higher pitch max), laugh more, use more *I* and *well*, use repair questions but not other kinds of questions, use more sexual terms, use far less appreciations and backchannels, and use fewer, longer turns, with more words in general. Features of the alter (the man) that helped our system detect women who say they are flirting include the male use of *you*, questions, and faster and quieter speech.

We also summarize here the features for the perception classification task; predicting which people will be labeled by their dates as flirting. Here the task is the same as for Jurafsky et al. (2009) and the values are similar.

• Men who are labeled by their female date as flirting present many of the same linguistic behaviors as when they express their intention to flirt. Some of the largest differences are that men are perceived to flirt when they use less appreciations and overlap less, while these features were not significant for men who said they were flirting. We also found that fast speech and more questions are more important features for flirtation perception than intention.

• Women who are labeled by their male date as flirting also present much of the same linguistic behavior as women who intend to flirt. Laughter, repair questions, and taking fewer, longer turns were not predictors of women labeled as flirting, although these were strong predictors of women intending to flirt.

Both genders convey intended flirtation by laughing more, speaking faster, and using higher pitch. However, we do find gender differences; men ask more questions when they say they are flirting, women ask fewer, although they do use more repair questions, which men do not. Women use more "I" and less "we"; men use more "we" and "you". Men labeled as flirting are softer, but women labeled as flirting are not. Women flirting use much fewer appreciations; appreciations were not a significant factor in men flirting.

8 Human Performance on this task

To evaluate the performance of our classifiers we compare against human labeled data.

We used the same test set as for our machine classifier; recall that this was created by taking the top ten percent of Likert ratings of the speaker's intention ratings by gender and called those positive for flirtation intention. We constructed negative examples by taking the bottom ten percent of intention Likert ratings. We called the interlocutor correct on the positive examples if the interlocutor's rating was greater than 5. Symmetrically for the negative examples, we said the interlocutor was correct if their rating was less than or equal to 5. Note that this metric is biased somewhat toward the humans and against our systems, because we do not penalize for intermediate values, while the system is trained to make binary predictions only on extremes. The results of the human perceivers on classifying flirtation intent are shown in Table 6.

Male speaker (Female perceiver)	Female speaker (Male perceiver)
62.2%	56.2%

Table 6: Accuracy of human listeners at labeling speakers as flirting or not.

We were quite surprised by the poor quality of the human results. Our system outperforms both men's performance in detecting women flirters (system 71.5% versus human 56.2%) and also women's performance in detecting male flirters (system 69.0% versus human 62.2%).

Why are humans worse than machines at detecting flirtation? We found a key insight by examining how the participants in a date label themselves and each other. Table 7 shows the 1-10 Likert values for the two participants in one of the dates, between Male 101 and Female 127. The two participants clearly had very different perspectives on the date. More important, however, we see that each participant labels their own flirting (almost) identically with their partner's flirting.

	I am flirting	Other is flirting
Male 101 says:	8	7
Female 127 says:	1	1

Table 7: Likert scores for the date between Female 127 and Male 101.

We therefore asked whether speakers in general tend to assign similar values to their own flirting and their partner's flirting. The Pearson correlation coefficient between these two variables (my perception of my own flirting, and my perception of other's flirting) is .73. By contrast, the poor performance of subjects at detecting flirting in their partners is coherent with the lower (.15) correlation coefficient between those two variables (my perception of the other's flirting, and the other's perception of their own flirting). This discrepancy is summarized in boldface in Table 8.

Since the speed-date data was also labeled for three other variables, we then asked the same question about these variables. As Table 8 shows, for all four styles, speakers' perception of others is strongly correlated with the speakers' perception of themselves, far more so than with what the others actually think they are doing.[3]

[3]This was true no matter how the correlations were run, whether with raw Likert values, with ego-centered (transformed) values and with self ego-centered but other raw.

Variable	Self-perceive-Other & Self-perceive-Self	Self-perceive-Other & Other-perceive-Other
Flirting	**.73**	**.15**
Friendly	.77	.05
Awkward	.58	.07
Assertive	.58	.09

Table 8: Correlations between speaker intentions and perception for all four styles.

Note that although perception of the other does not correlate highly with the other's intent for any of the styles, the correlations are somewhat better (.15) for flirting, perhaps because in the speed-date setting speakers are focusing more on detecting this behavior (Higgins and Bargh, 1987). It is also possible that for styles with positive valence (friendliness and flirting) speakers see more similarity between the self and the other than for negative styles (awkward and assertive) (Krahé, 1983).

Why should this strong bias exist to link self-flirting with perception of the other? One possibility is that speakers are just not very good at capturing the intentions of others in four minutes. Speakers instead base their judgments on their own behavior or intentions, perhaps because of a bias to maintain consistency in attitudes and relations (Festinger, 1957; Taylor, 1970) or to assume there is reciprocation in interpersonal perceptions (Kenny, 1998).

9 Conclusion

We have presented a new system that is able to predict flirtation intention better than humans can, despite humans having access to vastly richer information (visual features, gesture, etc.). This system facilitates the analysis of human perception and human interaction and provides a framework for understanding why humans perform so poorly on intention prediction.

At the heart of our system is a core set of prosodic, dialogue, and lexical features that allow for accurate prediction of both flirtation intention and flirtation perception. Since previous word lists don't capture sufficient lexical information, we used an autoencoder to automatically capture new lexical cues. The autoencoder shows potential for being a promising feature extraction method for social tasks where cues are domain specific.

Acknowledgments: Thanks to the anonymous reviewers and to a Google Research Award for partial funding.

References

J. Ang, R. Dhillon, A. Krupski, E. Shriberg, and A. Stolcke. 2002. Prosody-Based Automatic Detection of Annoyance and Frustration in Human-Computer Dialog. In *INTERSPEECH-02*.

Paul Boersma and David Weenink. 2005. Praat: doing phonetics by computer (version 4.3.14). [Computer program]. Retrieved May 26, 2005, from `http://www.praat.org/`.

M. A. Cohn, M. R. Mehl, and J. W. Pennebaker. 2004. Linguistic markers of psychological change surrounding September 11, 2001. *Psychological Science*, 15:687–693.

Leon Festinger. 1957. *A Theory of Cognitive Dissonance*. Row, Peterson, Evanston, IL.

E. Tory Higgins and John A. Bargh. 1987. Social cognition and social perception. *Annual Review of Psychology*, 38:369–425.

G. E. Hinton and R. R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

G. E. Hinton, S. Osindero, and Y. Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554.

Dan Jurafsky, Rajesh Ranganath, and Dan McFarland. 2009. Extracting social meaning: Identifying interactional style in spoken conversation. In *NAACL HLT 2009*, Boulder, CO.

David Kenny. 1998. *Interpersonal Perception: A Social Relations Analysis*. Guilford Press, New York, NY.

B. Krahé. 1983. Self-serving biases in perceived similarity and causal attributions of other people's performance. *Social Psychology Quarterly*, 46:318–329.

C. M. Lee and Shrikanth S. Narayanan. 2002. Combining acoustic and language information for emotion recognition. In *ICSLP-02*, pages 873–876, Denver, CO.

Jackson Liscombe, Jennifer Venditti, and Julia Hirschberg. 2003. Classifying Subject Ratings of Emotional Speech Using Acoustic Features. In *INTERSPEECH-03*.

Anmol Madan, Ron Caneel, and Alex Pentland. 2005. Voices of attraction. Presented at Augmented Cognition, HCI 2005, Las Vegas.

François Mairesse and Marilyn Walker. 2008. Trainable generation of big-five personality styles through data-driven parameter estimation. In *ACL-08*, Columbus.

François Mairesse, Marilyn Walker, Matthias Mehl, and Roger Moore. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of Artificial Intelligence Research (JAIR)*, 30:457–500.

Andrew Y. Ng. 2004. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *ICML 2004*.

J. W. Pennebaker and T. C. Lay. 2002. Language use and personality during crises: Analyses of Mayor Rudolph Giuliani's press conferences. *Journal of Research in Personality*, 36:271–282.

J. W. Pennebaker, R.E. Booth, and M.E. Francis. 2007. Linguistic inquiry and word count: LIWC2007 operator's manual. Technical report, University of Texas.

Alex Pentland. 2005. Socially aware computation and communication. *Computer*, pages 63–70.

Irina Rish. 2001. An empirical study of the naive bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*.

Andrew Rosenberg and Julia Hirschberg. 2005. Acoustic/prosodic and lexical correlates of charismatic speech. In *EUROSPEECH-05*, pages 513–516, Lisbon, Portugal.

S. S. Rude, E. M. Gortner, and J. W. Pennebaker. 2004. Language use of depressed and depression-vulnerable college students. *Cognition and Emotion*, 18:1121–1133.

B. Scholkopf, K.K. Sung, CJC Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. 1997. Comparing support vector machines with Gaussian kernels to radialbasis function classifiers. *IEEE Transactions on Signal Processing*, 45(11):2758–2765.

Howard Taylor. 1970. Chapter 2. In *Balance in Small Groups*. Von Nostrand Reinhold Company, New York, NY.

Recognizing Implicit Discourse Relations in the Penn Discourse Treebank

Ziheng Lin, Min-Yen Kan and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
{linzihen,kanmy,nght}@comp.nus.edu.sg

Abstract

We present an implicit discourse relation classifier in the Penn Discourse Treebank (PDTB). Our classifier considers the context of the two arguments, word pair information, as well as the arguments' internal constituent and dependency parses. Our results on the PDTB yields a significant 14.1% improvement over the baseline. In our error analysis, we discuss four challenges in recognizing implicit relations in the PDTB.

1 Introduction

In the field of discourse modeling, it is widely agreed that text is not understood in isolation, but in relation to its context. One focus in the study of discourse is to identify and label the relations between textual units (clauses, sentences, or paragraphs). Such research can enable downstream natural language processing (NLP) such as summarization, question answering, and textual entailment. For example, recognizing causal relations can assist in answering *why* questions. Detecting contrast and restatements is useful for paraphrasing and summarization systems. While different discourse frameworks have been proposed from different perspectives (Mann and Thompson, 1988; Hobbs, 1990; Lascarides and Asher, 1993; Knott and Sanders, 1998; Webber, 2004), most admit these basic types of discourse relationships between textual units.

When there is a discourse connective (e.g., *because*) between two text spans, it is often easy to recognize the relation between the spans, as most connectives are unambiguous (Miltsakaki et al., 2005; Pitler et al., 2008). On the other hand, it is difficult to recognize the discourse relations when there are no explicit textual cues. We term these cases explicit and implicit relations, respectively. While the recognition of discourse structure has been studied in the context of explicit relations (Marcu, 1998) in the past, little published work has yet attempted to recognize implicit discourse relations between text spans.

Detecting implicit relations is a critical step in forming a discourse understanding of text, as many text spans do not mark their discourse relations with explicit cues. Recently, the Penn Discourse Treebank (PDTB) has been released, which features discourse level annotation on both explicit and implicit relations. It provides a valuable linguistic resource towards understanding discourse relations and a common platform for researchers to develop discourse-centric systems. With the recent release of the second version of this corpus (Prasad et al., 2008), which provides a cleaner and more thorough implicit relation annotation, there is an opportunity to address this area of work.

In this paper, we provide classification of implicit discourse relations on the second version of the PDTB. The features we used include contextual modeling of relation dependencies, features extracted from constituent parse trees and dependency parse trees, and word pair features. We show an accuracy of 40.2%, which is a significant improvement of 14.1% over the majority baseline.

After reviewing related work, we first give an overview of the Penn Discourse Treebank. We then describe our classification methodology, followed by experimental results. We give a detailed discussion on the difficulties of implicit relation classification in the PDTB, and then conclude the paper.

2 Related Work

One of the first works that use statistical methods to detect implicit discourse relations is that of Marcu and Echihabi (2002). They showed that word pairs extracted from two text spans provide clues for detecting the discourse relation between

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 343–351,
Singapore, 6-7 August 2009.

the text spans. They used a set of textual patterns to automatically construct a large corpus of text span pairs from the web. These text spans were assumed to be instances of specific discourse relations. They removed the discourse connectives from the pairs to form an implicit relation corpus. From this corpus, they collected word pair statistics, which were used in a Naïve Bayes framework to classify discourse relations.

Saito et al. (2006) extended this theme, to show that phrasal patterns extracted from a text span pair provide useful evidence in the relation classification. For example, the pattern "... should have done ..." usually signals a contrast. The authors combined word pairs with phrasal patterns, and conducted experiments with these two feature classes to recognize implicit relations between adjacent sentences in a Japanese corpus.

Both of these previous works have the shortcoming of downgrading explicit relations to implicit ones by removing the explicit discourse connectives. While this is a good approach to automatically create large corpora, natively implicit relations may be signaled in different ways. The fact that explicit relations are explicitly signaled indicates that such relations need a cue to be unambiguous to human readers. Thus, such an artificial implicit relation corpus may exhibit marked differences from a natively implicit one. We validate this claim later in this work.

Wellner et al. (2006) used multiple knowledge sources to produce syntactic and lexico-semantic features, which were then used to automatically identify and classify explicit and implicit discourse relations in the Discourse Graphbank (Wolf and Gibson, 2005). Their experiments show that discourse connectives and the distance between the two text spans have the most impact, and event-based features also contribute to the performance. However, their system may not work well for implicit relations alone, as the two most prominent features only apply to explicit relations: implicit relations do not have discourse connectives and the two text spans of an implicit relation are usually adjacent to each other.

The work that is most related to ours is the forthcoming paper of Pitler et al. (2009) on implicit relation classification on the second version of the PDTB. They performed classification of implicit discourse relations using several linguistically informed features, such as word polarity, verb classes, and word pairs, showing performance increases over a random classification baseline.

3 Overview of the Penn Discourse Treebank

The Penn Discourse Treebank (PDTB) is a discourse level annotation (Prasad et al., 2008) over the one million word *Wall Street Journal* corpus. The PDTB adopts the predicate-argument view of discourse relations, where a discourse connective (e.g., *because*) is treated as a predicate that takes two text spans as its arguments. The argument that the discourse connective structurally attaches to is called Arg2, and the other argument is called Arg1. The PDTB provides annotations for explicit and implicit discourse relations. By definition, an explicit relation contains an explicit discourse connective. In the PDTB, 100 explicit connectives are annotated. Example 1 shows an explicit Contrast relation signaled by the discourse connective *but*. The last line shows the relation type and the file in the PDTB from which the example is drawn.

(1) **Arg1:** In any case, the brokerage firms are clearly moving faster to create new ads than they did in the fall of 1987.
Arg2: But it remains to be seen whether their ads will be any more effective.
(Contrast - wsj_2201)

In the PDTB, implicit relations are constrained by *adjacency*: only pairs of adjacent sentences within paragraphs are examined for the existence of implicit relations. When an implicit relation was inferred by an annotator, he/she inserted an implicit connective that best reflects the relation. Example 2 shows an implicit relation, where the annotator inferred a Cause relation and inserted an implicit connective *so* (i.e., the original text does not include *so*). The text in the box (*he says*) shows the attribution, i.e., the agent that expresses the arguments. The PDTB provides annotation for the attributions and supplements of the arguments.

(2) **Arg1:** "A lot of investor confidence comes from the fact that they can speak to us," he says .
Arg2: [**so**] "To maintain that dialogue is absolutely crucial."
(Cause - wsj_2201)

The PDTB provides a three level hierarchy of relation tags for its annotation. The first level consists of four major relation *classes*: Temporal, Contingency, Comparison, and Expansion. For each class, a second level of *types* is defined to provide finer semantic distinctions. A third level of *subtypes* is defined for only some types to specify the semantic contribution of each argument. Relation classes and types in the PDTB are reproduced in the first two columns of Table 1.

We focus on implicit relation classification of the Level 2 types in the PDTB, as we feel that Level 1 classes are too general and coarse-grained for downstream applications, while Level 3 subtypes are too fine-grained and are only provided for some types. Table 1 shows the distribution of the 16 Level 2 relation types of the implicit relations from the training sections, i.e., Sections 2 – 21. As there are too few training instances for Condition, Pragmatic Condition, Pragmatic Contrast, Pragmatic Concession, and Exception, we removed these five types from further consideration. We thus use the remaining 11 Level 2 types in our work. The initial distribution and adjusted distribution are shown in the last two columns of the table. We see that the three predominant types are Cause (25.63%), Conjunction (22.25%), and Restatement (19.23%).

Level 1 Class	Level 2 Type	Training instances	%	Adjusted %
Temporal	Asynchronous	583	4.36	4.36
	Synchrony	213	1.59	1.59
Contingency	Cause	3426	25.61	25.63
	Pragmatic Cause	69	0.52	0.52
	Condition	1	0.01	–
	Pragmatic Condition	1	0.01	–
Comparison	Contrast	1656	12.38	12.39
	Pragmatic Contrast	4	0.03	–
	Concession	196	1.47	1.47
	Pragmatic Concession	1	0.01	–
Expansion	Conjunction	2974	22.24	22.25
	Instantiation	1176	8.79	8.80
	Restatement	2570	19.21	19.23
	Alternative	158	1.18	1.18
	Exception	2	0.01	–
	List	345	2.58	2.58
Total		13375		
Adjusted total		13366		

Table 1: Distribution of Level 2 relation types of implicit relations from the training sections (Sec. 2 – 21). The last two columns show the initial distribution and the distribution after removing the five types that have only a few training instances.

4 Methodology

Our implicit relation classifier is built using supervised learning on a maximum entropy classifier. As such, our approach processes the annotated argument pairs into binary feature vectors suitable for use in training a classifier. Attributions and supplements are ignored from the relations, as our system does not make use of them. We chose the following four classes of features as they represent a wide range of information – contextual, syntactic, and lexical – that have been shown to be helpful in previous works and tasks. We now discuss the four categories of features used in our framework.

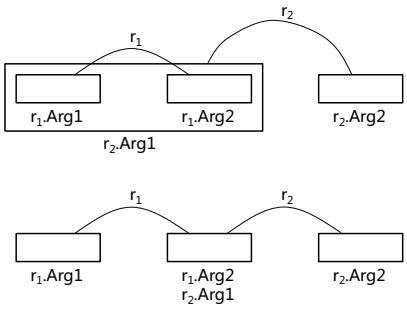


Figure 1: Two types of discourse dependency structures. Top: fully embedded argument, bottom: shared argument.

Contextual Features. Lee et al. (2006) showed that there are a variety of possible dependencies between pairs of discourse relations: independent, fully embedded argument, shared argument, properly contained argument, pure crossing, and partially overlapping argument. They argued that the last three cases – properly contained argument, pure crossing, and partially overlapping argument – can be factored out by appealing to discourse notions such as anaphora and attribution. Moreover, we also observed from the PDTB corpus that fully embedded argument and shared argument are the most common patterns, which are shown in Figure 1. The top portion of Figure 1 shows a case where relation r_1 is fully embedded in Arg1 of relation r_2, and the bottom portion shows r_1 and r_2 sharing an argument. We model these two patterns as contextual features. We believe that these discourse dependency patterns between a pair of adjacent relations are useful in identifying the relations. For example, if we have three items in a list, according to the PDTB binary predicate-argument definitions, there will be a List relation between

the first item and the second item, and another List relation between the previous List relation and the third item, where the previous List relation is fully embedded in Arg1 of the current List relation. As we are using the gold standard argument segmentation from the PDTB, we can extract and leverage these dependency patterns. For each relation *curr*, we use the previous relation *prev* and the next relation *next* as evidence to fire six binary features, as defined in Table 2.

Note that while *curr* is an implicit relation to be classified, both *prev* and *next* can be implicit or explicit relations. Pitler et al. (2008) showed that the type of a relation sometimes correlates to the type of its adjacent relation. When the adjacent relation is explicit, its type may be suggested by its discourse connective. Thus we include another two groups of contextual features representing the connectives of *prev* and *next* when they are explicit relations.

Fully embedded argument:
prev embedded in curr.Arg1
next embedded in curr.Arg2
curr embedded in prev.Arg2
curr embedded in next.Arg1
Shared argument:
prev.Arg2 = curr.Arg1
curr.Arg2 = next.Arg1

Table 2: Six contextual features derived from two discourse dependency patterns. *curr* is the relation we want to classify.

Constituent Parse Features. Research work from other NLP areas, such as semantic role labeling, has shown that features derived from syntactic trees are useful in semantic understanding. Such features include syntactic paths (Jiang and Ng, 2006) and tree fragments (Moschitti, 2004). From our observation of the PDTB relations, syntactic structure within one argument may constrain the relation type and the syntactic structure of the other argument. For example, the constituent parse structure in Figure 2(a) usually signals an Asynchronous relation when it appears in Arg2, as shown in Example 3, while the structure in Figure 2(b) usually acts as a clue for a Cause relation when it appears in Arg1, as shown in Example 4. In both examples, the lexicalized parts of the parse structure are bolded.

(3) **Arg1:** But the RTC also requires "working" capital to maintain the bad assets of thrifts that are sold
Arg2: [**subsequently**] That debt would be paid off **as** the assets are sold
(Asynchronous - wsj_2200)

(4) **Arg1:** It would **have been** too late to think about on Friday.
Arg2: [**so**] We had to think about it ahead of time.
(Cause - wsj_2201)

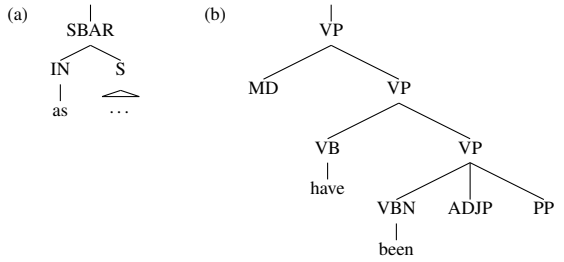


Figure 2: (a) constituent parse in Arg2 of Example 3, (b) constituent parse in Arg1 of Example 4.

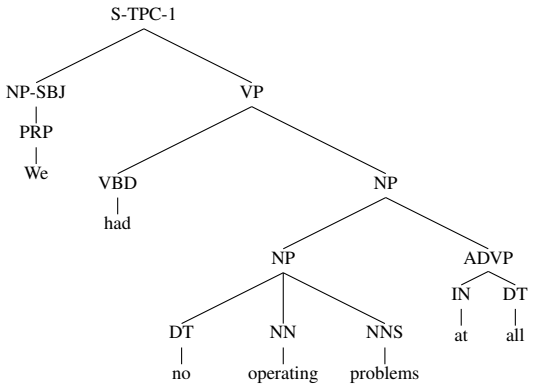


Figure 3: A gold standard subtree for Arg1 of an implicit discourse relation from wsj_2224.

For Arg1 and Arg2 of each relation, we extract the corresponding gold standard syntactic parse trees from the corpus. As an argument can be a single sentence, a clause, or multiple sentences, this results in a whole parse tree, parts of a parse tree, or multiple parse trees. From these parses, we extract all possible production rules. Although the structures shown in Figure 2 are tree fragments, tree fragments are not extracted as production rules act as generalization of tree fragments. As an example, Figure 3 shows the parse tree for Arg1 of an implicit discourse relation from the text wsj_2224. As Arg1 is a clause, the extracted tree

is a subtree. We then collect all production rules from this subtree, with function tags (e.g., SBJ) removed from internal nodes. POS tag to word production rules are collected as well. The resulting production rules include ones such as: S → NP VP, NP → PRP, PRP → "We", etc. Each production rule is represented as three binary features to check whether this rule appears in Arg1, Arg2, and both arguments.

Dependency Parse Features. We also experimented with features extracted from dependency trees of the arguments. We used the Stanford dependency parser (de Marneffe et al., 2006), which takes in a constituent parse tree and produces a dependency tree. Again, for an argument, we may collect a whole dependency tree, parts of a tree, or multiple trees, depending on the span of the argument. The reason for using dependency trees is that they encode additional information at the word level that is not explicitly present in the constituent trees. From each tree, we collect all words with the dependency types from their dependents. Figure 4 shows the dependency subtree for the same example in Figure 3, from which we collect three dependency rules: "had" ← nsubj dobj, "problems" ← det nn advmod, "at" ← dep.

Note that unlike the constituent parse features which are guaranteed to be accurate (as they are extracted from the gold parses of the corpus), the dependency parses occasionally contain errors. As with the constituent parse features, each dependency rule is represented as three binary features to check whether it appears in Arg1, Arg2, and both arguments.

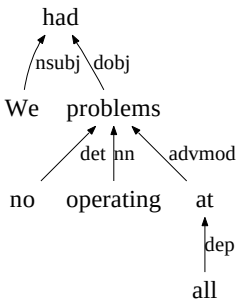


Figure 4: A dependency subtree for Arg1 of an implicit discourse relation from wsj_2224.

Lexical Features. Marcu and Echihabi (2002) demonstrated that word pairs extracted from the respective text spans are a good signal of the discourse relation between arguments. Thus we also consider word pairs as a feature class. We stemmed and collected all word pairs from Arg1 and Arg2, i.e., all (w_i, w_j) where w_i is a word from Arg1 and w_j a word from Arg2. Unlike their study, we limit the collection of word pair statistics to occurrences only in the PDTB corpus.

4.1 Feature Selection

For the collection of production rules, dependency rules, and word pairs, we used a frequency cutoff of 5 to remove infrequent features. From the implicit relation dataset of the training sections (i.e., Sec. 2 – 21), we extracted 11,113 production rules, 5,031 dependency rules, and 105,783 word pairs in total. We applied mutual information (MI) to these three classes of features separately, resulting in three ranked lists. A feature f has 11 MI values with all 11 types (for example, $MI(f, Cause)$ and $MI(f, Restatement)$), and we used the MI with the highest value for a feature to select features. In our experiments, the top features from the lists are used in the training and test phases.

5 Experiments

We experimented with a maximum entropy classifier from the OpenNLP MaxEnt package using various combinations of features to assess their efficacy. We used PDTB Sections 2 – 21 as our training set and Section 23 as the test set, and only used the implicit discourse relations.

In the PDTB, about 2.2% of the implicit relations are annotated with two types, as shown in Example 7 in Section 6. During training, a relation that is annotated with two types is considered as two training instances, each with one of the types. During testing, such a relation is considered one test instance, and if the classifier assigns either of the two types, we consider it as correct. Thus, the test accuracy is calculated as the number of correctly classified test instances divided by the total number of test instances.

In our work, we use the majority class as the baseline, where all instances are classified as Cause. This yields an accuracy of 26.1% on the test set. A random baseline yields an even lower accuracy of 9.1% on the test set.

5.1 Results and Analysis

To check the efficacy of the different feature classes, we trained individual classifiers on all features within a single feature class (Rows 1 to 4 in Table 3) as well as a single classifier trained

with all features from all feature classes (Row 5). Among the four individual feature classes, production rules and word pairs yield significantly better performance over the baseline with $p < 0.01$ and $p < 0.05$ respectively, while context features perform slightly better than the baseline.

	# Production rules	# Dependency rules	# Word pairs	Context	Acc.
R1	11,113	–	–	No	36.7%
R2	–	5,031	–	No	26.0%
R3	–	–	105,783	No	30.3%
R4	–	–	–	Yes	28.5%
R5	11,113	5,031	105,783	Yes	35.0%

Table 3: Classification accuracy with all features from each feature class. Rows 1 to 4: individual feature class; Row 5: all feature classes.

Interestingly, we noted that the performance with all dependency rules is slightly lower than the baseline (Row 2), and applying all feature classes does not yield the highest accuracy (Row 5), which we suspected were due to noise. To confirm this, we employed MI to select the top 100 production rules and dependency rules, and the top 500 word pairs (as word pairs are more sparse). We then repeated the same set of experiments, as shown in Table 4 (Row 4 of this table is repeated from Table 3 for consistency). With only the top features, production rules, dependency rules, and word pairs all gave significant improvement over the baseline with $p < 0.01$. When we used all feature classes, as in the last row, we obtained the highest accuracy of 40.2%.

	# Production rules	# Dependency rules	# Word pairs	Context	Acc.
R1	100	–	–	No	38.4%
R2	–	100	–	No	32.4%
R3	–	–	500	No	32.9%
R4	–	–	–	Yes	28.5%
R5	100	100	500	Yes	40.2%

Table 4: Classification accuracy with top rules/word pairs for each feature class. Rows 1 to 4: individual feature class; Row 5: all feature classes.

Table 4 also validates the pattern of predictiveness of the feature classes: production rules contribute the most to the performance individually, followed by word pairs, dependency rules, and finally, context features. A natural question to ask is whether any of these feature classes can be omitted to achieve the same level of performance as the combined classifier. To answer this question, we conducted a final set of experiments, in which we gradually added in feature classes in the order of their predictiveness (i.e., production rules $\succ$ word pairs $\succ$ dependency rules $\succ$ context features), with results shown in Table 5. These results confirm that each additional feature class indeed contributes a marginal performance improvement, (although it is not significant) and that all feature classes are needed for optimal performance.

	# Production rules	# Dependency rules	# Word pairs	Context	Acc.
R1	100	–	–	No	38.4%
R2	100	–	500	No	38.9%
R3	100	100	500	No	39.0%
R4	100	100	500	Yes	40.2%

Table 5: Accuracy with feature classes gradually added in the order of their predictiveness.

Note that Row 3 of Table 3 corresponds to Marcu and Echihabi (2002)'s system which applies only word pair features. The difference is that they used a Naïve Bayes classifier while we used a maximum entropy classifier. As we did not implement their Naïve Bayes classifier, we compare their method's performance using the result from Table 3 Row 3 with ours from Table 5 Row 4, which shows that our system significantly ($p < 0.01$) outperforms theirs.

Level 2 Type	Precision	Recall	F_1	Count in test set
Asynchronous	0.50	0.08	0.13	13
Synchrony	–	–	–	5
Cause	0.39	0.76	0.51	200
Pragmatic Cause	–	–	–	5
Contrast	0.61	0.09	0.15	127
Concession	–	–	–	5
Conjunction	0.30	0.51	0.38	118
Instantiation	0.67	0.39	0.49	72
Restatement	0.48	0.27	0.35	190
Alternative	–	–	–	15
List	0.80	0.13	0.23	30
All (Micro Avg.)	0.40	0.40	0.40	780

Table 6: Recall, precision, F_1, and counts for 11 Level 2 relation types. "–" indicates 0.00.

Table 6 shows the recall, precision, and F_1 measure for the 11 individual Level 2 relation types in the final experiment set up (Row 4 from Table 5). A point worth noting is that the classifier labels no instances of the Synchrony, Pragmatic Cause, Concession, and Alternative relation types. The reason is that the percentages for these four types are so small that the classifier is highly skewed towards the other types. From the distribution shown in Table 1, there are just 4.76% training data for these four types, but 95.24% for the remaining seven types. In fact, only 30 test instances are labeled with these four types, as shown in the last column of Table 6. As Cause is the most pre-

dominant type in the training data, the classifier tends to label uncertain relations as Cause, thus giving Cause high recall but low precision. We see that the F measures correlate well with the training data frequency, thus we hypothesize that accuracy may improve if more training data for low frequency relations can be provided.

Our work differs from that of (Pitler et al., 2009) in that our system performs classification at the more fine-grained Level 2 types, instead of the coarse-grained Level 1 classes. Their system applies a Naïve Bayes classifier whereas our system uses a maximum entropy classifier, and the sets of features used are also different. In addition, the data set of (Pitler et al., 2009) includes `EntRel` and `AltLex`, which are relations in which an implicit connective cannot be inserted between adjacent sentences, whereas ours excludes `EntRel` and `AltLex`.

6 Discussion: Why are implicit discourse relations difficult to recognize?

In the above experiments, we have shown that by using the four feature classes, we are able to increase the classification accuracy from 26.1% of the majority baseline to 40.2%. Although we feel a 14.1 absolute percentage improvement is a solid result, an accuracy of 40% does not allow downstream NLP applications to trust the output of such a classification system.

To understand the difficulties of the task more deeply, we analyzed individual training and validation data pairs, from which we were able to generalize four challenges to automated implicit discourse relation recognition. We hope that this discussion may motivate future work on implicit discourse relation recognition.

Ambiguity. There is ambiguity among the relations. For example, we notice that a lot of Contrast relations are mistakenly classified as Conjunction. When we analyzed these relations, we observed that Contrast and Conjunction in the PDTB annotation are very similar to each other in terms of words, syntax, and semantics, as Examples 5 and 6 show. In both examples, the same antonymous verb pair is used (*fell* and *rose*), different subjects are mentioned in Arg1 and Arg2 (*net* and *revenue* in the first example, and *net* and *sales* in the second), and these subjects are all compared to like items from the previous year. Moreover, the implicit discourse connective given by the annotators is *while* in both cases, which is an ambiguous connective as shown in (Miltsakaki et al., 2005).

(5) **Arg1:** In the third quarter, AMR said, net fell to $137 million, or $2.16 a share, from $150.3 million, or $2.50 a share.
Arg2: [**while**] Revenue rose 17% to $2.73 billion from $2.33 billion a year earlier.
(Contrast - wsj_1812)

(6) **Arg1:** Dow's third-quarter net fell to $589 million, or $3.29 a share, from $632 million, or $3.36 a share, a year ago.
Arg2: [**while**] Sales in the latest quarter rose 2% to $4.25 billion from $4.15 billion a year earlier.
(Conjunction - wsj_1926)

Relation ambiguity may be ameliorated if an instance is analyzed in context. However, according to the PDTB annotation guidelines, if the annotators could not disambiguate between two relation types, or if they felt both equally reflect their understanding of the relation between the arguments, they could annotate two types to the relation. In the whole PDTB corpus, about 5.4% of the explicit relations and 2.2% of the implicit relations are annotated with two relation types. Example 7 is such a case where the implicit connective *meanwhile* may be interpreted as expressing a Conjunction or Contrast relation.

(7) **Arg1:** Sales surged 40% to 250.17 billion yen from 178.61 billion.
Arg2: [**meanwhile**] Net income rose 11% to 29.62 billion yen from 26.68 billion.
(Conjunction; Contrast - wsj_2242)

Inference. Sometimes inference and a knowledge base are required to resolve the relation type. In Example 8, to understand that Arg2 is a restatement of Arg1, we need a semantic mechanism to show that either the semantics of Arg1 infers that of Arg2 or the other way around. In the below example, *I had calls all night long* infers *I was woken up every hour* semantically, as shown in: $receive\_call(I) \wedge duration(all\_night) \Rightarrow woken\_up(I) \wedge duration(every\_hour)$.

(8) **Arg1:** "I had calls all night long from the States," he said.
Arg2: "[**in fact**] I was woken up every hour – 1:30, 2:30, 3:30, 4:30."
(Restatement - wsj_2205)

In fact, most relation types can be represented using formal semantics (PDTB-Group, 2007), as shown in Table 7, where $|Arg1|$ and $|Arg2|$ represent the semantics extracted from Arg1 and Arg2, respectively. This kind of formal semantic reasoning requires a robust knowledge base, which is still beyond our current technology.

Relation type	Semantic representation
Cause	$\|Arg1\| \prec \|Arg2\| \quad \vee \quad \|Arg2\| \prec \|Arg1\|$
Concession	$A \prec C \quad \wedge \quad B \Rightarrow \neg C$ where $A \in \|Arg1\|, B \in \|Arg2\|$
Instantiation	$exemplify(\|Arg2\|, \lambda x.x \in E)$ where $E = extract(\|Arg1\|)$
Restatement	$\|Arg1\| \Rightarrow \|Arg2\| \quad \vee \quad \|Arg1\| \Leftarrow \|Arg2\|$
Alternative	$\|Arg1\| \wedge \|Arg2\| \quad \vee \quad \|Arg1\| \oplus \|Arg2\|$

Table 7: Some examples of relation types with their semantic representations, as taken from (PDTB-Group, 2007).

Context. PDTB annotators adopted the *Minimality Principle* in argument selection, according to which they only included in the argument the minimal span of text that is sufficient for the interpretation of the relation. While the context is not necessary to interpret the relation, it is usually necessary to understand the meaning of the arguments. Without an analysis of the context, Arg1 and Arg2 may seem unconnected, as the following example shows, where the meaning of Arg1 is mostly derived from its previous context (i.e., *West German ... technical reactions*).

(9) **Prev. Context:** West German Economics Minister Helmut Haussmann said, "In my view, the stock market will stabilize relatively quickly. There may be one or other psychological or technical reactions,
Arg1: but they aren't based on fundamentals.
Arg2: [**in short**] The economy of West Germany and the EC European Community is highly stable."
(Conjunction - wsj_2210)

Sometimes the range of the context may easily extend to the whole text, which would require a system to possess a robust context modeling mechanism. In Example 10, in order to realize the causal relation between Arg2 and Arg1, we possibly need to read the whole article and understand what was happening: the machinist union was having a strike and the strike prevented most of its union members from working.

(10) **Arg1:** And at the company's Wichita, Kan., plant, about 2,400 of the 11,700 machinists still are working, Boeing said.
Arg2: [**because**] Under Kansas right-to-work laws, contracts cannot require workers to be union members.
(Cause - wsj_2208)

World Knowledge. Sometimes even context modeling is not enough. We may also need world knowledge to understand the arguments and hence to interpret the relation. In the following example, from the previous sentence of Arg1, it is reported that "the Senate voted to send a delegation of congressional staffers to Poland to assist its legislature", and this delegation is viewed as a "gift" in Arg1. It is suggested in Arg2 that the Poles might view the delegation as a "Trojan Horse". Here we need world knowledge to understand that "Trojan Horse" is usually applied as a metaphor for a person or thing that appears innocent but has harmful intent, and hence understand that Arg2 poses a contrasting view of the delegation as Arg1 does.

(11) **Arg1:** Senator Pete Domenici calls this effort "the first gift of democracy".
Arg2: [**but**] The Poles might do better to view it as a Trojan Horse.
(Contrast - wsj_2237)

These four classes of difficulties – ambiguity between relations, inference, contextual modeling, and world knowledge – show that implicit discourse relation classification needs deeper semantic representations, more robust system design, and access to more external knowledge. These obstacles may not be restricted to recognizing implicit relations, but are also applicable to other related discourse-centric tasks.

7 Conclusion

We implemented an implicit discourse relation classifier and showed initial results on the recently released Penn Discourse Treebank. The features we used include the modeling of the context of relations, features extracted from constituent parse trees and dependency parse trees, and word pair features. Our classifier achieves an accuracy of 40.2%, a 14.1% absolute improvement over the baseline. We also conducted a data analysis and discussed four challenges that need to be addressed in future to overcome the difficulties of implicit relation classification in the PDTB.

References

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, pages 449–454.

Jerry R. Hobbs. 1990. Literature and cognition. In *CSLI Lecture Notes Number 21*. CSLI Publications.

Zheng Ping Jiang and Hwee Tou Ng. 2006. Semantic role labeling of NomBank: A maximum entropy approach. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 138–145, Sydney, Australia.

Alistair Knott and Ted Sanders. 1998. The classification of coherence relations and their linguistic markers: An exploration of two languages. *Journal of Pragmatics*, 30(2):135–175.

Alex Lascarides and Nicholas Asher. 1993. Temporal interpretation, discourse relations and commonsense entailment. *Linguistics and Philosophy*, 16(5):437–493.

Alan Lee, Rashmi Prasad, Aravind Joshi, Nikhil Dinesh, and Bonnie Webber. 2006. Complexity of dependencies in discourse: Are dependencies in discourse more complex than in syntax? In *Proceedings of the 5th International Workshop on Treebanks and Linguistic Theories*, Prague, Czech Republic, December.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 368–375, Morristown, NJ, USA.

Daniel Marcu. 1998. A surface-based approach to identifying discourse markers and elementary textual units in unrestricted texts. In *Proceedings of the COLING-ACL 1998 Workshop on Discourse Relations and Discourse Markers*, pages 1–7, Montreal, Canada, August.

Eleni Miltsakaki, Nikhil Dinesh, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2005. Experiments on sense annotations and sense disambiguation of discourse connectives. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT2005)*, Barcelona, Spain, December.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, Barcelona, Spain.

PDTB-Group, 2007. *The Penn Discourse Treebank 2.0 Annotation Manual*. The PDTB Research Group, December.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind Joshi. 2008. Easily identifiable discourse relations. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, Manchester, UK, August.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. To appear in *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2009)*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse Treebank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*.

Manami Saito, Kazuhide Yamamoto, and Satoshi Sekine. 2006. Using phrasal patterns to identify discourse relations. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2006)*, pages 133–136, New York, USA, June.

Bonnie Webber. 2004. D-LTAG: Extending lexicalized TAG to discourse. *Cognitive Science*, 28(5):751–779, September.

Ben Wellner, James Pustejovsky, Catherine Havasi, Anna Rumshisky, and Roser Sauri. 2006. Classification of discourse coherence relations: An exploratory study using multiple knowledge sources. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, Sydney, Australia, July.

Florian Wolf and Edward Gibson. 2005. Representing discourse coherence: a corpus-based analysis. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 134–140, Morristown, NJ, USA.

A Bayesian Model of Syntax-Directed Tree to String Grammar Induction

Trevor Cohn and **Phil Blunsom**
School of Informatics
University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
Scotland, United Kingdom
{tcohn,pblunsom}@inf.ed.ac.uk

Abstract

Tree based translation models are a compelling means of integrating linguistic information into machine translation. Syntax can inform lexical selection and reordering choices and thereby improve translation quality. Research to date has focussed primarily on decoding with such models, but less on the difficult problem of inducing the bilingual grammar from data. We propose a generative Bayesian model of tree-to-string translation which induces grammars that are both smaller and produce better translations than the previous heuristic two-stage approach which employs a separate word alignment step.

1 Introduction

Many recent advances in statistical machine translation (SMT) are a result of the incorporation of syntactic knowledge into the translation process (Marcu et al., 2006; Zollmann and Venugopal, 2006). This has been facilitated by the use of synchronous grammars to model translation as a generative process over pairs of strings in two languages. Such models are particularly attractive for translating between languages with divergent word orders, such as Chinese and English, where syntax-inspired translation rules can succinctly describe the requisite reordering operations. In contrast, standard phrase-based models (Koehn et al., 2003) assume a mostly monotone mapping between source and target, and therefore cannot adequately model these phenomena. Currently the most successful paradigm for the use of synchronous grammars in translation is that of string-to-tree transduction (Galley et al., 2004; Zollmann and Venugopal, 2006; Galley et al., 2006; Marcu et al., 2006). In this case a grammar is extracted from a parallel corpus, with strings on its source side and syntax trees on its target side, which is then used to translate novel sentences by performing inference over the space of target syntax trees licensed by the grammar.

To date grammar-based translation models have relied on heuristics to extract a grammar from a word-aligned parallel corpus. These heuristics are extensions of those developed for phrase-based models (Koehn et al., 2003), and involve symmetrising two directional word alignments followed by a projection step which uses the alignments to find a mapping between source words and nodes in the target parse trees (Galley et al., 2004). However, such approaches leave much to be desired. Word-alignments rarely factorise cleanly with parse trees (i.e., alignment points cross constituent structures), resulting in large and implausible translation rules which generalise poorly to unseen data (Fossum et al., 2008). The principal reason for employing a grammar based formalism is to induce rules which capture long-range reorderings between source and target. However if the grammar itself is extracted using word-alignments induced with models that are unable to capture such reorderings, it is unlikely that the grammar will live up to expectations.

In this work we draw on recent advances in Bayesian modelling of grammar induction (Johnson et al., 2007; Cohn et al., 2009) to propose a non-parametric model of synchronous tree substitution grammar (STSG), continuing a recent trend in SMT to seek principled probabilistic formulations for heuristic translation models (Zhang et al., 2008; DeNero et al., 2008; Blunsom et al., 2009b; Blunsom et al., 2009a). This model leverages a hierarchical Bayesian prior to induce a compact translation grammar directly from a parsed parallel corpus, unconstrained by word-alignments. We show that the induced grammars are more plausible and improve translation output.

This paper is structured as follows: In Section

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 352–361,
Singapore, 6-7 August 2009.

2 we introduce the STSG formalism and describe current heuristic approaches to grammar induction. We define a principled Bayesian model of string-to-tree translation in Section 3, and describe an inference technique using Gibbs sampling in Section 4. In Section 5 we analyse an induced grammar on a corpus of Chinese→English translation, comparing them with a heuristic grammar in terms of grammar size and translation quality.

2 Background

Current tree-to-string translation models are a form of *Synchronous Tree Substitution Grammar* (STSG; Eisner (2003)). Formally, a STSG is a 5-tuple, $G = (T, T', N, S, R)$, where T and T' are sets of *terminal symbols* in the target and source languages respectively, N is a set of *non-terminal symbols*, $S \in N$ is the distinguished *root non-terminal* and R is a set of productions (a.k.a. rules). Each production is a tuple comprising an *elementary tree* and a string, the former referring to a tree fragment of depth ≥ 1 where each internal node is labelled with a non-terminal and each leaf is labelled with either a terminal or a non-terminal. The string part of the rule describes the lexical component of the rule in the source language and includes a special variable for each *frontier non-terminal* in the elementary tree. These variables describe the reordering and form the recursion sites in the generative process of creating tree and string pairs with the grammar. For example, the rule

$$\langle (\text{NP NP}_{\boxed{1}} \text{ (PP (IN of) NP}_{\boxed{2}})), \boxed{2}\ 的\ \boxed{1} \rangle \quad (1)$$

rewrites a noun-phrase (NP) as a NP and prepositional phrase (PP) headed by 'of' in the target language. The rule generates the token '的' in the source and reverses the order of the two child noun-phrases, indicated by the numbering of the variables in the string part.

A *derivation* creates a (tree, string) pair by starting with the root non-terminal and an empty string, then choosing a rule to rewrite (substitute) the non-terminal and expand the string. This process repeats by rewriting all frontier non-terminals until there are none remaining. A *Probabilistic STSG* assigns a probability to each rule in the grammar. The probability of a derivation is the product of the probabilities of its component rules, and the probability of a (tree, string) pair is the sum of the probabilities over all its derivations.

2.1 Heuristic Grammar Induction

Grammar based SMT models almost exclusively follow the same two-stage approach to grammar induction developed for phrase-based methods (Koehn et al., 2003). In this approach they induce a finite-state grammar with phrase-pairs as rules by taking a sentence aligned parallel corpus and 1) predicting word alignments before 2) extracting transduction rules that are 'consistent' with the word aligned data. Although empirically effective, this two stage approach is less than ideal due to the disconnect between the word-based models used for alignment and the phrase-based translation model. This is problematic as the word-based model cannot recognise phrase-based phenomena. Moreover, it raises the problem of identifying and weighting the rules from the word alignment.

The same criticisms levied at the phrase-based models apply equally to the two-stage technique used for synchronous grammar induction (Galley et al., 2004; Zollmann and Venugopal, 2006; Galley et al., 2006; Marcu et al., 2006). Namely that the word alignment models typically do not use any syntax and therefore will not be able to model, e.g., consistent syntactic reordering effects, or the impact of the syntactic category on phrase translations. The identification and estimation of grammar rules from word aligned data is also non-trivial. Galley et al. (2004) describe an algorithm for inducing a string-to-tree grammar using a parallel corpus with syntax trees on target side. Their method projects the source strings onto nodes of the target tree using the word alignment, and then extracts the minimal transduction rules as well as rules composed of adjacent minimal units. The production weights are estimated either by heuristic counting (Koehn et al., 2003) or using the EM algorithm. Both estimation techniques are flawed. The heuristic method is inconsistent in the limit (Johnson, 2002) while EM is *degenerate*, placing disproportionate probability mass on the largest rules in order to describe the data with as few a rules as possible (DeNero et al., 2006). With no limit on rule size this method will learn a single rule for every training instance, and therefore will not generalise to unseen sentences. These problems can be ameliorated by imposing limits on rule size or early stopping of EM training, however neither of these techniques addresses the underlying problems.

In contrast, our model is trained in a single step, i.e., the alignment model *is* the translation model. This allows syntax to directly inform the alignments. We infer a grammar without resorting to word alignment constraints or limits on rule size. The model uses a prior to bias towards a compact grammar with small rules, thus solving the degeneracy problem.

3 Model

Our training data comprises parallel target trees and source strings and our aim is to induce a STSG that best describes this data. This is achieved by inferring a distribution over the derivations for each training instance, where the set of derivations collectively specify the grammar. In the following, we denote the source trees as $\mathbf{t}$, target strings $\mathbf{s}$, and derivations $\mathbf{r}$ which are sequences of grammar rules, r.

As described in section 2.1, previous methods for estimating a STSG have suffered from degeneracy. A principled way to correct for such degenerated behaviour is to use a *prior* over rules which biases towards small rules. This matches our intuition: we expect good translation rules to be small, with few internal nodes, frontier non-terminals and terminal strings. However, we recognise that on occasion larger rules will be necessary; we allow such rules when there is sufficient support in the data.

We model the grammar as a set of distributions, G_c, over the productions for each non-terminal symbol, c. We adopt a non-parametric Bayesian approach by treating each G_c as a random variable with a Dirichlet process (DP) prior,

$$\begin{aligned} r|c &\sim G_c \\ G_c|\alpha_c, P_0 &\sim \text{DP}(\alpha_c, P_0(\cdot|c))\,, \end{aligned}$$

where $P_0(\cdot|c)$ (the *base distribution*) is a distribution over the infinite space of trees rooted with c, and α_c (the *concentration parameter*) controls the model's tendency towards either reusing existing rules or creating novel ones as each training instance is encountered (and consequently, the tendency to infer larger or smaller grammars). We discuss the base distribution in more detail below.

Rather than representing the distribution G_c explicitly, we integrate over all possible values of G_c. This leads to the following predictive distribution for the rule r_i given the previously observed rules $\mathbf{r}^{-i} = r_1 \dots r_{i-1}$,

$$p(r_i|\mathbf{r}^{-i}, c, \alpha_c, P_0) = \frac{n_{r_i}^{-i} + \alpha_c P_0(r_i|c)}{n_c^{-i} + \alpha_c}\,, \quad (2)$$

where $n_{r_i}^{-i}$ is the number number of times r_i has been used to rewrite c in $\mathbf{r}^{-i}$, and $n_c^{-i} = \sum_{r,R(r)=c} n_r^{-i}$ is the total count of rewriting c (here $R(r)$ is the root non-terminal of r). The distribution is *exchangeable*, meaning that all permutations of the input sequence are assigned the same probability. This allows us to treat any item as being the last, which is fundamental for efficient Gibbs sampling. Henceforth we adopt the notation $\mathbf{r}^-$ and n^- to refer to the rules and counts for the whole data set excluding the current rules under consideration, irrespective of their location in the corpus.

The base distribution, P_0, assigns a prior probability to an infinite number of rules, where each rule is an (elementary tree, source string) pair denoted $r = (\mathbf{e}, \mathbf{w})$. While there are a myriad of possible distributions, we developed a very simple one. We decompose the probability into two factors,

$$P_0(\mathbf{e}, \mathbf{w}|c) = P(\mathbf{e}|c)P(\mathbf{w}|\mathbf{e})\,, \quad (3)$$

the probability of the target elementary tree and the probability of the source string, where $c = R(r)$.

The tree probability, $P(\mathbf{e}|c)$ in (3), is modelled using generative process whereby the root category c is expanded into a sequence of child non-terminals, then each of these are either expanded or left as-is. This process continues until there are no unprocessed children. The number of child nodes for each expansion is drawn from a geometric prior with parameter p_{child}, except in the case of pre-terminals where the number of children is always one. The binary expansion decisions are drawn from a Bernoulli prior with parameter p_{expand}, and non-terminals and terminals are drawn uniformly from N and T respectively. For example, the source side of rule (1) was generated as follows: 1) the NP was rewritten as two children; 2) an NP; and 4) a PP; 5) the NP child was not expanded; 6) the PP child was expanded; 7) as an IN; and 8) a NP; 9) the IN was expanded to the terminal 'of'; and 10) the final NP was not expanded. Each of these steps is a draw from the relevant distribution, and the total probability is the product of the probabilities for each step.

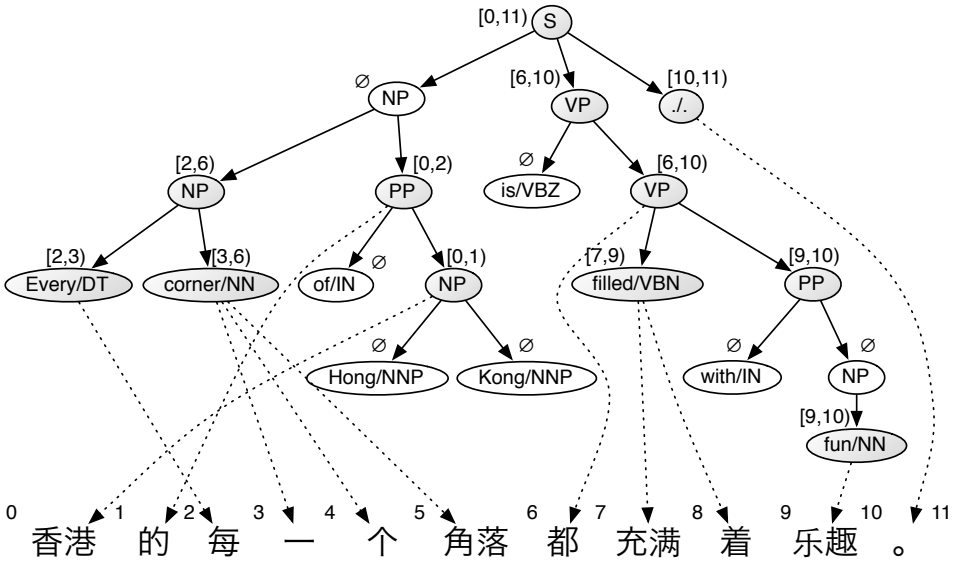


Figure 1: Example derivation. Each node is annotated with their span in the target string (aligned nodes are shaded). The dotted edges show the implied alignments. Preterminals are displayed with their child terminal in the leaf nodes.

The second factor in (3) is $P(\mathbf{w}|\mathbf{e})$, the probability of the source string (a sequence of source terminals and variables). We assume that the elementary tree is generated first, and condition the string probability on $l = F(\mathbf{e})$, its number of frontier nodes (i.e., variables). The string is then created by choosing a number of terminals from a geometric prior with parameter p_{term} then drawing each terminal from a uniform distribution over T'. Finally each of the l variables are inserted into the string one at a time using a uniform distribution over the possible placements. For the example rule in (1) the generative process corresponds to 1) deciding to create one terminal; 2) with value 的; 3) inserting the first variable after the terminal; and 4) inserting the second variable before the terminal. Again, the probability of the string is simply the product of the probabilities for each step.

Together both the factors in the base distribution penalise large trees with many nodes and long strings with many terminals and variables. P_0 decreases exponentially with rule size, thus discouraging the model from using larger rules; for this to occur the rules must significantly increase the likelihood.

4 Training

To train our model we use Gibbs sampling (Geman and Geman, 1984), a Markov chain Monte Carlo method (Gilks et al., 1996) in which variables are repeatedly sampled conditioned on the values of all other variables in the model.[1] After a period of burn-in, each sampler state (set of variable assignments) is a sample from the posterior distribution of the model. In our case, we wish to sample from the posterior over the grammar, $P(\mathbf{r}|\mathbf{t}, \mathbf{s}, \alpha)$.

To simplify matters we associate an alignment variable, a, with every internal node of the trees in the training set. This variable specifies the span of source tokens to which node is aligned. Alternatively, the node can be unaligned, which is encoded as an empty span. I.e. $a \in (J \times J) \cup \emptyset$ where J is the set of target word indices. Spans are written $[i, j)$: inclusive of i and exclusive of j. Each aligned node ($a \neq \emptyset$) forms the root of a rule as well as being a frontier non-terminal of an ancestor rule, while unaligned nodes form part of an ancestor rule.[2] The set of valid alignments are constrained by the tree in a number of ways. Child nodes can be aligned only to subspans of their ancestor nodes' alignments and no two nodes' alignments can overlap. Finally, the root node of the tree must be aligned to the full

[1] Previous approaches to bilingual grammar induction have used variational inference to optimise a bound on the data log-likelihood (Zhang et al., 2008; Blunsom et al., 2009b). Both these approaches truncated the grammar a priori in order to permit tractable inference. In contrast our Gibbs sampler can perform inference over the full space of grammars. See also Blunsom et al. (2009a) where we present a Gibbs sampler for inducing SCFGs without truncation.

[2] The Gibbs sampler is an extension of our sampler for monolingual tree-substitution grammar (Cohn et al., 2009), which used a binary substitution variable at each node to encode the segmentation of a training tree into elementary trees.

⟨(S (NP NP[1] PP[2]) VP[3] .[4]), [2] [1] [3] [4]⟩
⟨(NP DT[1] NN[2]), [1] [2]⟩
⟨(DT Every), 每⟩
⟨(NN corner), 一 个 角落⟩
⟨(PP (IN of) NP[1]), [1] 的⟩
⟨(NP (NNP Hong) (NNP Kong)), 香港⟩
⟨(VP (VBZ is) VP[1]), [1]⟩
⟨(VP VBN[1] PP[2]), 都 [1] [2]⟩
⟨(VBN filled), 充 着⟩
⟨(PP (IN with) (NP NN[1])), [1]⟩
⟨(NN fun), 趣⟩
⟨(. .), 。⟩

Table 1: Grammar rules specified by the derivation in Figure 1. Each rule is shown as a tuple comprising a target elementary tree and a source string. Boxed numbers show the alignment between string variables and frontier nonterminals.

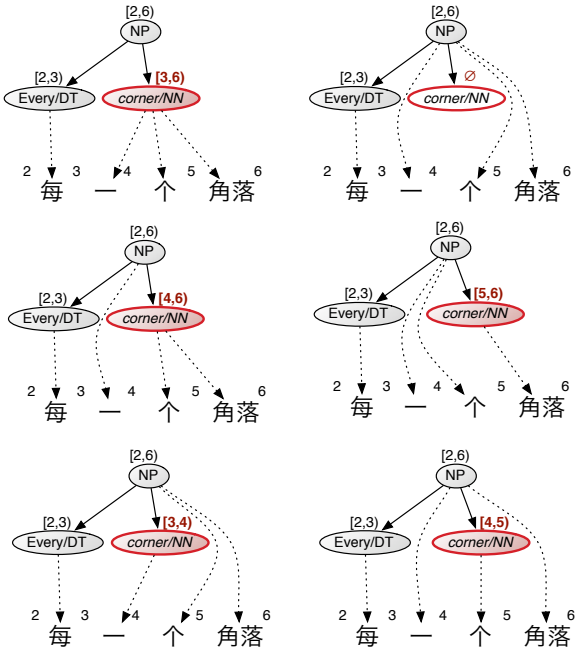


Figure 2: Possible state updates for the (NN corner) node using the EXPAND operator.

span of source words.

Collectively, the training trees and alignment variables specify the sequence of rules $\mathbf{r}$, which in turn specify the grammar. Figure 1 shows an example derivation with alignment variables. The corresponding STSG rules are shown in Table 1.

4.1 Gibbs operators

The Gibbs sampler works by sampling new values of the alignment variables, using two different Gibbs operators to make the updates. The first operator, EXPAND, takes a tree node, v, and samples a new alignment, a_v, given the alignments of all other nodes in the same tree and all other trees in the corpus, denoted $\mathbf{a}^-$. The set of valid labels is constrained by the other alignments in the tree, specifically that of the node's closest aligned ancestor, a_p, its closest aligned descendants, $\mathbf{a}_d$, and its aligned siblings, $\mathbf{a}_s$ (the aligned descendants of a). The alignment variable may be empty, $a_v = \emptyset$, while non-empty values must obey the tree constraints. Specifically the span must be a subspan of its ancestor, $a_v \subseteq a_p$, subsume its descendants, $a_v \supseteq \bigcup \mathbf{a}_d$, and not overlap its siblings, $j \notin \bigcup \mathbf{a}_s, \forall j \in a_v$. Figure 2 shows an example with the range of valid values for *corner/NN*'s alignment variable and the corresponding alignments that these encode.

Each alignment in the set of valid outcomes defines a set of grammar rules. The non-aligned outcome results in a single rule r_p rooted at ancestor node p. While the various aligned outcomes result in a pair of rules, $r_{p'}$ and r_v, rooted at p and v respectively. In the example in Figure 2, the top-right outcome has $a_v = \emptyset$ and

$$r_p = \langle (\text{NP DT}_{[1]}\ (\text{NN corner})), [1]\ 一\ 个\ 角落 \rangle .$$

The bottom-right outcome, $a_v = [4, 5)$, describes the pair of rules:

$$r_{p'} = \langle (\text{NP DT}_{[1]}\ \text{NN}_{[2]}), [1]\ 一\ [2]\ 角落 \rangle \text{ and}$$
$$r_v = \langle (\text{NN corner}), 个 \rangle .$$

The set of valid options are then scored according to the probability of their rules as follows:

$$P(r_p|\mathbf{r}^-) = \frac{n^-_{r_p} + \alpha P_0(r_p|c_p)}{n^-_{c_p} + \alpha} \quad (4)$$

$$\begin{aligned} P(r_{p'}, r_v|\mathbf{r}^-) &= P(r_{p'}|\mathbf{r}^-)P(r_v|\mathbf{r}^-, r_{p'}) \\ &= \frac{n^-_{r_{p'}} + \alpha P_0(r_{p'}|c_p)}{n^-_{c_p} + \alpha} \times \\ &\quad \frac{n^-_{r_v} + \delta(r_{p'}, r_v) + \alpha P_0(r_v|c_v)}{n^-_{c_v} + \delta(c_p, c_v) + \alpha} \end{aligned} \quad (5)$$

where c_p is the non-terminal at node p (similarly for c_v), n^- denote counts of trees (e.g., $n^-_{r_p}$) or the sum over all trees expanding a non-terminal (e.g., $n^-_{c_v}$) in the conditioning context, $\mathbf{r}^-$, and $\delta(\cdot, \cdot)$ is the Kronecker delta function, which returns 1 when its arguments are identical and 0 otherwise. For clarity, we have omitted some items from the conditioning context

in (4) and (5), namely $\mathbf{t}, \mathbf{s}$ and hyper-parameters $\alpha, p_{\text{child}}, p_{\text{expand}}, p_{\text{term}}$. The δ terms in the second factor of (5) account for the changes to n^- that would occur after observing $r_{p'}$, which forms part of the conditioning context for r_v. If the rules $r_{p'}$ and r_v are identical, then the count $n^-_{r_v}$ would increase by one, and if the rules expand the same root non-terminal, then $n^-_{c_v}$ would increase by one. Equation (4) is evaluated once for the unaligned outcome, $a_v = \emptyset$, and (5) is evaluated for each valid alignment. The probabilities are normalised and an outcome sampled.

The EXPAND operator is sufficient to move from one derivation to any other valid derivation, however it may take many steps to do so. These intermediate steps may require the sampler to pass through highly improbable regions of the state space, and consequently such moves are unlikely. The second operator, SWAP, is designed to help address this problem by increasing the mobility of the sampler, allowing it to mix more quickly. The operator considers pairs of nodes, v, w, in one tree and attempts to swap their alignment values.[3] This is illustrated in the example in Figure 3. There are two options being compared: preserving the alignments (left) or swapping them (right). This can change three rules implied by the derivation: that rooted at the nodes' common aligned ancestor, p, and those rooted at v and w. For the example, the left option implies rules

$$\begin{aligned}\{r_p &= \langle(\text{NP DT}\boxed{1}\ \text{NN}\boxed{2}), \boxed{1}\ \boxed{2}\rangle,\\ r_v &= \langle(\text{DT Every}), 每\rangle,\\ r_w &= \langle(\text{NN corner}), 一\ 个\ 角落\rangle\}\,,\end{aligned}$$

and the right option implies rules

$$\begin{aligned}\{r_p &= \langle(\text{NP DT}\boxed{1}\ \text{NN}\boxed{2}), \boxed{2}\ \boxed{1}\rangle,\\ r_v &= \langle(\text{DT Every}), 一\ 个\ 角落\rangle,\\ r_w &= \langle(\text{NN corner}), 每\rangle\}\,.\end{aligned}$$

We simply evaluate the probability of both triples of rules under our model, $P(r_p, r_v, r_w|\mathbf{r}^-) = P(r_p|\mathbf{r}^-)P(r_v|\mathbf{r}^-, r_p)$ $P(r_w|\mathbf{r}^-, r_p, r_v)$, where the additional rules in the conditioning context signify their inclusion in the counts $\mathbf{r}^-$ before applying (2) to evaluate the probability (much the same as in (5) where the δ functions encode the changes to the counts). An outcome is then sampled according to the normalised probabilities of the preserve vs. swap rules.

[3] We rarely need to consider the full quadratic space of node pairs, as the validity constraints mean that the only candidates for swapping are siblings (i.e., share the closest aligned ancestor) which do not have any aligned descendants.

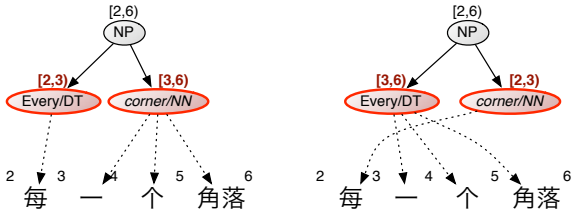


Figure 3: Possible state updates for the pair of nodes (DT every) and (NN corner) using the SWAP operator.

	English ← Chinese	
Sentences	300k	
Words or Segments	11.0M	8.6M
Avg. Sent. Length	36	28
Longest Sent.	80	80

Table 2: NIST Chinese-English corpora statistics (LDC2003E14, LDC2005E47).

The Gibbs sampler makes use of both operators. The algorithm visits each (tree, string) pair in the training set in random order and applies the EXPAND operator to every node in the tree. After the tree has been processed, the SWAP operator is applied to all candidate pairs of nodes. Visiting all sentence pairs in this way constitutes a single sample from the Gibbs sampler.

5 Experiments

We evaluate our non-parametric model of grammar induction on a subset of the NIST Chinese-English translation evaluation, representing a realistic SMT experiment with millions of words and long sentences. The Chinese-English training data consists of the FBIS corpus (LDC2003E14) and the first 100k sentence pairs from the Sinorama corpus (LDC2005E47). The Chinese text was segmented with a CRF-based Chinese segmenter optimized for MT (Chang et al., 2008), and the English text was parsed using the Stanford parser (Klein and Manning, 2003).

As a baseline we implemented the heuristic grammar extraction technique of Galley et al. (2004) (henceforth GHKM). This method finds the minimum sized translation rules which are consistent with a word-aligned sentence pair, as

described in section 2.1. The rules are then treated as events in a relative frequency estimate.[4] We used Giza++ Model 4 to obtain word alignments (Och and Ney, 2003), using the `grow-diag-final-and` heuristic to symmetrise the two directional predictions (Koehn et al., 2003).

The model was sampled for 300 iterations to 'burn-in', where in each iteration we applied both sampling operators to all nodes (or node pairs) of all training instances. We initialised the sampler using the GHKM derivation of the training data (the baseline system). The final state of the sampler was used to extract the grammar. The hyperparameters were set by hand to $\alpha = 10^6$, $p_{\text{child}} = 0.5$, $p_{\text{expand}} = 0.5$, and $p_{\text{term}} = 0.5$.[5] Overall the model took on average 2,218s per full iteration of Gibbs sampling and 1 week in total to train, using a single core of a 2.3Ghz AMD Opteron machine.

5.1 Grammar Analysis

The resulting grammar had 1.62M rules, almost identical to the GHKM grammar which had 1.63M. Despite their similarity in size the grammars were quite different, as illustrated in Figure 4, which shows histograms over various measures of rule size for the two grammars. Under each measure the sampled grammar finds many more simple rules – shallower with fewer internal nodes, fewer variables and fewer terminals – than the GHKM method. This demonstrates that the prior is effective in shifting mass away from complex rules. To show how the rules themselves differ, Table 3 lists rules in the sampled grammar that are not in the GHKM grammar. Note that many of these rules are highly plausible, describing regular tree structures and lexicalisation. These rules have not been specified to the same extent in the GHKM grammar. For example the first rule incorporates

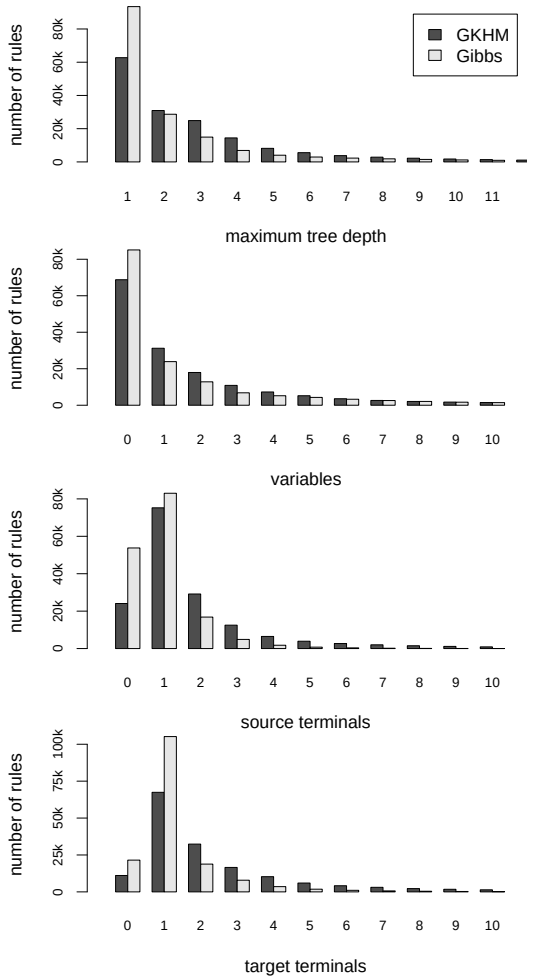


Figure 4: Histograms over rule statistics comparing the heuristic grammar (GHKM) and learnt grammar (Gibbs).

the TOP symbol, while the GHKM grammar instead relies on the rule $\langle(\text{TOP S}\boxed{1}), \boxed{1}\rangle$ to produce the same fragment. The model has learnt to distinguish between sentence-spanning and subsentential S constituents, which typically do not include final punctuation. The third and ninth (last) rule are particularly interesting. These rules encode reordering effects relating to noun phrases and subordinate prepositional phrases, in particular that Chinese prepositional modifiers precede the nouns they modify. Differences in word order such as these are quite common in Chinese-English corpora, so it is imperative that they are modelled accurately.

The rules in the GHKM grammar that do not appear in the sampled grammar are shown in Table 4. In contrast to the rules only present in the sampled grammar, these have much lower counts, i.e., are less probable. Each of these rules has been specified further by the Bayesian model.

[4] Our implementation of the GHKM algorithm attaches unaligned source words to the highest possible node in the source tree, rather than allowing all attachment points as in the original presentation (Galley et al., 2004). Allowing all attachments made no difference to translation performance, but did make the grammar considerably larger. We implemented only the minimal rule extraction, i.e., with no rule composition (Galley et al., 2006). Consequently there is no derivational ambiguity, obviating the need for expectation maximisation or similar for rule estimation.

[5] Note that although α seems large, it still encourages sparse distributions as the P_0 values are typically much smaller than its reciprocal, 10^{-6}, especially if the rule is large. $\alpha P_0 < 1$ implies a sparse Dirichlet prior.

⟨(TOP (S NP[1] VP[2] .[3])), [1] [2] [3]⟩
⟨(S (VP (TO to) VP[1])), [1]⟩
⟨(NP NP[1] (PP (IN of) NP[2])), [2] [1]⟩
⟨(PP (IN in) NP[1]), 在 [1]⟩
⟨(NP NP[1] (PP (IN of) NP[2])), [1] [2]⟩
⟨(NP (DT the) NN[1]), 的 [1]⟩
⟨(S (VP TO[1] VP[2])), [1] [2]⟩
⟨(VP (VBZ is) NP[1]), 是 [1]⟩
⟨(NP (NP (DT the) NN[1]) (PP (IN of) NP[2])), [2] [1]⟩

Table 3: Top ten rules in the sampled grammar that do not appear in the GHKM grammar. All the above rules are quite high probability, with counts between 37,118 and 7,275 from first to last.

⟨(PP (IN at) (NP DT[1] (NNS levels))), [1] 級⟩
⟨(NP NP[1] ,[2] NP[3] (, ,) CC[4] NP[5]), [1] [2] [3] [4] [5]⟩
⟨(NP NP[1] ,[2] NP[3] ,[4] NP[5] (, ,) (CC and) NP[6]), [1] [2] [3] [4] [5] , [6]⟩
⟨(S S[1] (NP (PRP They)) VP[2] .[3]), [1] [2] [3]⟩
⟨(S PP[1] ,[2] NP[3] VP[4] .[5] "[6]), [1] [2] [3] [4] [6] [5]⟩
⟨(S PP[1] ,[2] NP[3] VP[4] .[5]), [1] 中 [2] [3] [4] [5]⟩
⟨(NP (NNP Foreign) (NNP Ministry) NN[1] (NNP Zhu) (NNP Bangzao)), 外交部 [1] 朱邦造⟩
⟨(S S[1] S[2]), [1] [2]⟩
⟨(S S[1] (NP (PRP We)) VP[2] .[3]), [1] [2] [3]⟩
⟨(NP (DT the) (NNS people) POS[1]), 人民 [1]⟩

Table 4: Top ten rules in the GHKM grammar that do not appear in the sampled grammar. These are quite low probability rules: their counts range from 1,137 to 103.

For example, every instance of the first rule had the same determiner and target translation, ⟨(PP (IN at) (NP (DT all) (NNS levels))), 各 級⟩, and therefore the model specified the determiner, resulting in a single rule. The model has correctly learnt that other translations for (DT all) are not appropriate in this context (e.g., 都, 所有 or 一切). In a number of the remaining rules the commas were lexicalised, or S rules were extended to include the TOP symbol.

To further illustrate the differences between the grammars, Table 5 shows the rules which include the possessive particle, 的, and at least one variable. In both grammars there are many fully lexicalised rules which translate the token to, e.g., a determiner or a preposition. The grammars differ on the complex rules which combine lexicalisation and frontier non-terminals. The GHKM rules are all very simple depth-1 SCFG rules, containing minimal information. In contrast, the sampled rules are more lexicalised, licensing the insertion of various English tokens and tree substructure. Note particularly the second and forth rule which succinctly describe the reordering of prepositional phrases with an noun phrase.

Sampled Grammar
⟨(NP (DT the) NN[1]), 的 [1]⟩
⟨(NP (NP (DT the) NN[1]) (PP (IN of) NP[2])), [2] 的 [1]⟩
⟨(NP (DT the) NN[1]), [1] 的⟩
⟨(NP (NP (DT the) JJ[1] NN[2]) (PP (IN of) NP[3])), [3] 的 [1] [2]⟩
⟨(PP (IN of) NP[1]), [1] 的⟩
GHKM Grammar
⟨(NP JJ[1] NNS[2]), [1] 的 [2]⟩
⟨(NP JJ[1] NN[2]), [1] 的 [2]⟩
⟨(NP DT[1] JJ[2] NN[3]), [1] [2] 的 [3]⟩
⟨(NP PRP$[1] NN[2]), [1] 的 [2]⟩
⟨(NP NP[1] PP[2]), [2] 的 [1]⟩

Table 5: Top five rules which include the possessive particle 的 and at least one variable.

5.2 Translation

In order to test the translation performance of the grammars induced by our model and the GHKM method[6] we report BLEU (Papineni et al., 2002) scores on sentences of up to twenty words in length from the MT03 NIST evaluation. We built a synchronous beam search decoder to find the maximum scoring derivation, based on the CYK+ chart parsing algorithm and the cube-pruning method of Chiang (2007). Parse edges for all constituents spanning a given chart cell were cube-pruned together using a beam of width 1000, and only edges from the top ten constituents in each cell were retained. No artificial glue-rules or rule span limits were employed.[7] The parameters of the translation system were trained to maximize BLEU on the MT02 test set (Och, 2003). Decoding took roughly 10s per sentence for both grammars, using a 8-core 2.6Ghz Intel Xeon machine.

Table 6 shows the BLEU scores for the baseline using the GHKM rule induction algorithm, and our non-parametric Bayesian grammar induction method. We see a small increase in generalisation performance from our model. Our previous anal-

[6]Our decoder was unable to process unary rules (those which consume nothing in the source). Monolingual parsing with unary productions is fairly straightforward (Stolcke, 1995), however in the transductive setting these rules can licence infinite insertions in the target string. This is further complicated by the language model integration. Therefore we composed each unary rule instance with its descendant rule(s) to create a non-unary rule.

[7]Our decoder lacks certain features shown to be beneficial to synchronous grammar decoding, in particular rule binarisation (Zhang et al., 2006). As such the reported results for MT03 lag the state-of-the-art: the Moses phrase-based decoder (Koehn et al., 2007) achieves 26.8. We believe that improvements from a better decoder implementation would be orthogonal to the improvements presented here (and would allow us to relax the length restriction on the test set).

Model	BLEU score
GHKM	26.0
Our model	26.6

Table 6: Translation results on the NIST test set MT03 for sentences of length ≤ 20.

ysis (Section 5.1) of the grammars produced by the two approaches showed our method produced better lexicalised rules than those induced by the GHKM algorithm. Galley et al. (2006) noted that the GHKM algorithm often over generalised and proposed combining minimal rules to form composed rules as a solution. Although composing rules was effective at improving BLEU scores, the result was a massive expansion in the size of the grammar. By learning the appropriate level of lexicalisation we believe that our inference algorithm is having a similar effect as composing rules (Galley et al., 2006), however the resulting grammar remains compact, a significant advantage of our approach.

6 Conclusion

In this paper we have presented a method for inducing a tree-to-string grammar which removes the need for various heuristics and constraints from models of word alignment. Instead the model is capable of directly inferring a grammar in one step, using the syntactic fragments that it has learnt to better align the source and target data. Using a prior which favours sparse distributions and simpler rules, we demonstrate that the model finds a more parsimonious grammar than the heuristic technique. Moreover, this grammar results in improved translations on a standard evaluation set.

We expect that various extensions to the model would improve its performance. One avenue is to develop a more sophisticated prior over rules, e.g., one that recognises common types of rule via the shape of the tree and ordering pattern in the target. A second avenue is to develop better means of inference under the grammar, in order to ensure faster mixing and a means to escape from local optima. Finally, we wish to develop a method for decoding under the full Bayesian model, instead of the current beam search. With these extensions we expect that our model of grammar induction has the potential to greatly improve translation output.

Acknowledgements

The authors acknowledge the support of the EPSRC (grants GR/T04557/01 and EP/D074959/1). This work has made use of the resources provided by the Edinburgh Compute and Data Facility (ECDF). The ECDF is partially supported by the eDIKT initiative.

References

P. Blunsom, T. Cohn, C. Dyer, M. Osborne. 2009a. A Gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, Singapore. To appear.

P. Blunsom, T. Cohn, M. Osborne. 2009b. Bayesian synchronous grammar induction. In D. Koller, D. Schuurmans, Y. Bengio, L. Bottou, eds., *Advances in Neural Information Processing Systems 21*, 161–168. MIT Press, Cambridge, MA.

P.-C. Chang, M. Galley, C. D. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, 224–232, Columbus, Ohio.

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

T. Cohn, S. Goldwater, P. Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 548–556, Boulder, Colorado.

J. DeNero, D. Gillick, J. Zhang, D. Klein. 2006. Why generative phrase models underperform surface heuristics. In *Proceedings on the Workshop on Statistical Machine Translation*, 31–38, New York City, NY.

J. DeNero, A. Bouchard-Côté, D. Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 314–323, Honolulu, Hawaii.

J. Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, 205–208, Sapporo, Japan.

V. Fossum, K. Knight, S. Abney. 2008. Using syntax to improve word alignment precision for syntax-based machine translation. In *Proceedings of the*

Third Workshop on Statistical Machine Translation, 44–52, Columbus, Ohio.

M. Galley, M. Hopkins, K. Knight, D. Marcu. 2004. What's in a translation rule? In *Proceedings of the 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 273–280, Boston, MA.

M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, I. Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 961–968, Sydney, Australia.

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

W. Gilks, S. Richardson, D. J. Spiegelhalter, eds. 1996. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, Suffolk.

M. Johnson, T. L. Griffiths, S. Goldwater. 2007. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In B. Schölkopf, J. Platt, T. Hoffman, eds., *Advances in Neural Information Processing Systems 19*, 641–648. MIT Press, Cambridge, MA.

M. Johnson. 2002. The DOP estimation method is biased and inconsistent. *Computational Lingusitics*, 28(1):71–76.

D. Klein, C. D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *In Advances in Neural Information Processing Systems 15*, 3–10. MIT Press.

P. Koehn, F. J. Och, D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 48–54, Edmonton, Canada.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, 177–180, Prague, Czech Republic.

D. Marcu, W. Wang, A. Echihabi, K. Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, 44–52, Sydney, Australia.

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

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 160–167, Sapporo, Japan.

K. Papineni, S. Roukos, T. Ward, W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318, Philadelphia, PA.

A. Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2).

H. Zhang, L. Huang, D. Gildea, K. Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the 2006 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 256–263.

H. Zhang, C. Quirk, R. C. Moore, D. Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *Proceedings of ACL-08: HLT*, 97–105, Columbus, Ohio.

A. Zollmann, A. Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the 2006 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 138–141, New York City, NY.

Better Synchronous Binarization for Machine Translation

Tong Xiao[*], Mu Li[+], Dongdong Zhang[+], Jingbo Zhu[*], Ming Zhou[+]

[*]Natural Language Processing Lab
Northeastern University
Shenyang, China, 110004
xiaotong@mail.neu.edu.cn
zhujingbo@mail.neu.edu.cn

[+]Microsoft Research Asia
Sigma Center
Beijing, China, 100080
muli@microsoft.com
dozhang@microsoft.com
mingzhou@microsoft.com

Abstract

Binarization of Synchronous Context Free Grammars (SCFG) is essential for achieving polynomial time complexity of decoding for SCFG parsing based machine translation systems. In this paper, we first investigate the excess edge competition issue caused by a left-heavy binary SCFG derived with the method of Zhang et al. (2006). Then we propose a new binarization method to mitigate the problem by exploring other alternative equivalent binary SCFGs. We present an algorithm that iteratively improves the resulting binary SCFG, and empirically show that our method can improve a string-to-tree statistical machine translations system based on the synchronous binarization method in Zhang et al. (2006) on the NIST machine translation evaluation tasks.

1 Introduction

Recently Statistical Machine Translation (SMT) systems based on Synchronous Context Free Grammar (SCFG) have been extensively investigated (Chiang, 2005; Galley et al., 2004; Galley et al., 2006) and have achieved state-of-the-art performance. In these systems, machine translation decoding is cast as a synchronous parsing task. Because general SCFG parsing is an NP-hard problem (Satta and Peserico, 2005), practical SMT decoders based on SCFG parsing requires an equivalent binary SCFG that is directly learned from training data to achieve polynomial time complexity using the CKY algorithm (Kasami, 1965; Younger, 1967) borrowed from CFG parsing techniques. Zhang et al. (2006) proposed *synchronous binarization*, a principled method to binarize an SCFG in such a way that both the source-side and target-side virtual non-terminals have contiguous spans. This property of synchronous binarization guarantees the polynomial time complexity of SCFG parsers even when an *n*-gram language model is integrated, which has been proved to be one of the keys to the success of a string-to-tree syntax-based SMT system.

However, as shown by Chiang (2007), SCFG-based decoding with an integrated *n*-gram language model still has a time complexity of $\Theta(m^3|T|^{4(n-1)})$, where m is the source sentence length, and $|T|$ is the vocabulary size of the language model. Although it is not exponential in theory, the actual complexity can still be very high in practice. Here is an example extracted from real data. Given the following SCFG rule:

$$\boldsymbol{VP} \rightarrow \boldsymbol{VB}\ \boldsymbol{NP}\ 会\ \boldsymbol{JJR}\ ,\ \boldsymbol{VB}\ \boldsymbol{NP}\ \text{will be}\ \boldsymbol{JJR}$$

we can obtain a set of equivalent binary rules using the synchronous binarization method (Zhang et al., 2006) as follows:

$\boldsymbol{VP} \rightarrow \boldsymbol{V}_1\ \boldsymbol{JJR}$,	$\boldsymbol{V}_1\ \boldsymbol{JJR}$
$\boldsymbol{V}_1 \rightarrow \boldsymbol{VB}\ \boldsymbol{V}_2$,	$\boldsymbol{VB}\ \boldsymbol{V}_2$
$\boldsymbol{V}_2 \rightarrow \boldsymbol{NP}\ 会$,	$\boldsymbol{NP}$ will be

This binarization is shown with the solid lines as binarization (a) in Figure 1. We can see that binarization (a) requires that "NP 会" should be reduced at first. Data analysis shows that "NP 会" is a frequent pattern in the training corpus, and there are 874 binary rules of which the source language sides are "NP 会". Consequently these binary rules generate a large number of competing edges in the chart when "NP 会" is matched in decoding. To reduce the number of edges pro-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 362–370,
Singapore, 6-7 August 2009.

posed in decoding, hypothesis re-combination is used to combine the equivalent edges in terms of dynamic programming. Generally, two edges can be re-combined if they satisfy the following two constraints: 1) the LHS (left-hand side) non-terminals are identical and the sub-alignments are the same (Zhang et al., 2006); and 2) the boundary words[1] on both sides of the partial translations are equal between the two edges (Chiang, 2007). However, as shown in Figure 2, the decoder still generates 801 edges after the hypothesis re-combination. As a result, aggressive pruning with beam search has to be employed to reduce the search space to make the decoding practical. Usually in beam search only a very small number of edges are kept in the beam of each chart cell (e.g. less than 100). These edges have to compete with each other to survive from the pruning. Obviously, more competing edges proposed during decoding can lead to a higher risk of making search errors.

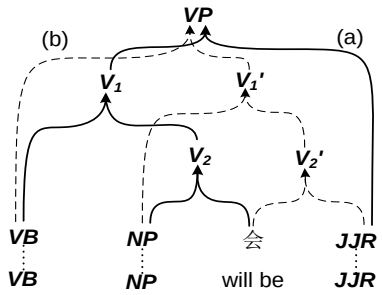


Figure 1: Two different binarizations (a) and (b) of the same SCFG rule distinguished by the solid lines and dashed lines

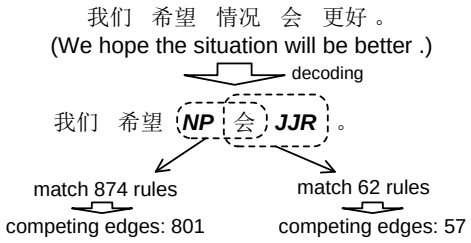


Figure 2: Edge competitions caused by different binarizations

The edge competition problem for SMT decoding is not addressed in previous work (Zhang et al., 2006; Huang, 2007) in which each SCFG rule is binarized in a fixed way. Actually the results of synchronous binarization may not be the only solution. As illustrated in Figure 1, the rule can also be binarized as binarization (b) which is shown with the dashed lines.

We think that this problem can be alleviated by choosing better binarizations for SMT decoders, since there is generally more than one binarization for a SCFG rule. In our investigation, about 96% rules that need to be binarized have more than one binarization under the contiguous constraint. As shown in binarization (b) (Figure 1), "会 JJR" is reduced first. In the decoder, the number of binary rules with the source-side "会 JJR" is 62, and the corresponding number of edges is 57 (Figure 2). The two numbers are both much smaller than those of "NP 会" in (a). This is an informative clue that the binarization (b) could be better than the binarization (a) based on the following: the probability of pruning the rule in (a) is higher than that in (b) as the rule in (b) has fewer competitors and has more chances to survive during pruning.

In this paper we propose a novel binarization method, aiming to find better binarizations to improve an SCFG-based machine translation system. We formulate the binarization optimization as a cost reduction process, where the cost is defined as the number of rules sharing a common source-side derivation in an SCFG. We present an algorithm, *iterative cost reduction* algorithm, to obtain better binarization for the SCFG learnt automatically from the training corpus. It can work with an efficient CKY-style binarizer to search for the lowest-cost binarization. We apply our method into a state-of-the-art string-to-tree SMT system. The experimental results show that our method outperforms the synchronous binarization method (Zhang et al., 2006) with over 0.8 BLEU scores on both NIST 2005 and NIST 2008 Chinese-to-English evaluation data sets.

2 Related Work

The problem of binarization originates from the parsing problem in which several binarization methods are studied such as left/right binarization (Charniak et al., 1998; Tsuruoka and Tsujii, 2004) and head binarization (Charniak et al., 2006). Generally, the pruning issue in SMT decoding is unnecessary for the parsing problem, and the accuracy of parsing does not rely on the binarization method heavily. Thus, many efforts on the binarization in parsing are made for the efficiency improvement instead of the accuracy improvement (Song et al., 2008).

Binarization is also an important topic in the research of syntax-based SMT. A synchronous

[1] For the case of *n*-gram language model integration, $2 \times (n-1)$ boundary words needs to be examined.

binarization method is proposed in (Zhang et al., 2006) whose basic idea is to build a left-heavy binary synchronous tree (Shapiro and Stephens, 1991) with a left-to-right shift-reduce algorithm. Target-side binarization is another binarization method which is proposed by Huang (2007). It works in a left-to-right way on the target language side. Although this method is comparatively easy to be implemented, it just achieves the same performance as the synchronous binarization method (Zhang et al., 2006) for syntax-based SMT systems. In addition, it cannot be easily integrated into the decoding of some syntax-based models (Galley et al., 2004; Marcu et al., 2006), because it does not guarantee contiguous spans on the source language side.

3 Synchronous Binarization Optimization by Cost Reduction

As discussed in Section 1, binarizing an SCFG in a fixed (left-heavy) way (Zhang et al., 2006) may lead to a large number of competing edges and consequently high risk of making search errors. Fortunately, in most cases a binarizable SCFG can be binarized in different ways, which provides us with an opportunity to find a better solution than the default left-heavy binarization. An ideal solution to this problem could be that we define an exact edge competition estimation function and choose the best binary SCFG based on it. However, even for the rules with a common source-side, generally it is difficult to estimate the exact number of competing edges in the dynamic SCFG parsing process for machine translation, because in order to integrate an *n*-gram language model, the actual number of edges not only depends on SCFG rules, but also depends on language model states which are specific to input sentences. Instead, we have to employ certain kinds of approximation of it. First we will introduce some notations frequently used in later discussions.

3.1 Notations

We use $G = \{R_i : X_i \rightarrow \alpha_i, \beta_i\}$ to denote an SCFG, where R_i is the i^{th} rule in G; X_i is the LHS (left hand side) non-terminal of R_i; α_i and β_i are the source-side and target-side RHS (right hand side) derivations of R_i respectively. We use $\mathcal{B}(G)$ to denote the set of equivalent binary SCFG of G. The goal of SCFG binarization is to find an appropriate binary SCFG $G' \in \mathcal{B}(G)$. For R_i, $\mathcal{B}(R_i) = \{v_{ij}\} \subseteq G' \in \mathcal{B}(G)$ is the set of equivalent binary rules based on R_i, where v_{ij} is the j^{th} binary rule in $\mathcal{B}(R_i)$. Figure 3 illustrates the meanings of these notations with a sample grammar.

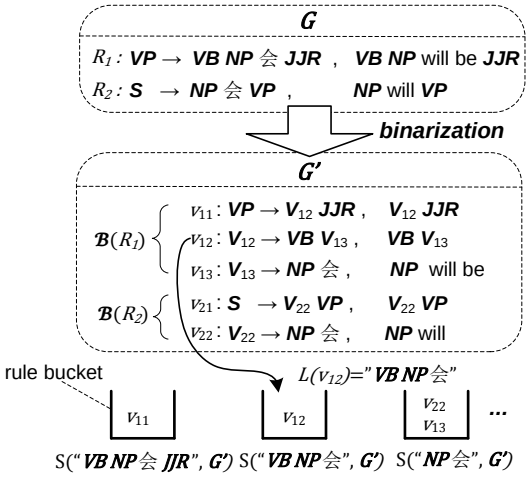


Figure 3: Binarization on a sample grammar

The function $L(\cdot)$ is defined to map a resulting binary rule $v_{ij} \epsilon G'$ to the sub-sequence in α_i derived from v_{ij}. For example, as shown in Figure 3, the binary rule v_{13} covers the source sub-sequence "NP 会" in R_1, so $L(v_{13}) =$ "NP 会". Similarly, $L(v_{12}) =$ "VB NP 会".

The function $L(\cdot)$ is used to group the rules in G' with a common right-hand side derivation for source language. Given a binary rule $v \in G'$, we can put it into a *bucket* in which all the binary rules have the same source sub-sequence $L(v)$. For example (Figure 3), as $L(v_{12}) =$ "VB NP 会", v_{12} is put into the bucket indexed by "VB NP 会". And v_{13} and v_{22} are put into the same bucket, since they have the same source sub-sequence "NP 会". Obviously, G' can be divided into a set of mutual exclusive rule buckets by $L(\cdot)$.

In this paper, we use $S(L(v), G')$ to denote the bucket for the binary rules having the source sub-sequence $L(v)$. For example, $S("NP\ 会", G')$ denotes the bucket for the binary rules having the source-side "NP 会". For simplicity, we also use $S(v, G')$ to denote $S(L(v), G')$.

3.2 Cost Reduction for SCFG Binarization

Given a binary SCFG G', it can be easily noticed that if a rule v in the bucket $S(v, G')$ can be applied to generate one or more new edges in SCFG parsing, any other rules in this bucket can also be applied because all of them can be reduced from the same underlying derivation $L(v)$.

Each application of other rules in the bucket $S(v, G')$ can generate competing edges with the one based on v. Intuitively, the size of bucket can be used to approximately indicate the actual number of competing edges on average, and reducing the size of bucket could help reduce the edges generated in a parsing chart by applying the rules in the bucket. Therefore, if we can find a method to greedily reduce the size of each bucket $S(v, G')$, we can reduce the overall expected edge competitions when parsing with G'.

However, it can be easily proved that the numbers of binary rules in any $G' \in \mathcal{B}(G)$ are same, which implies that we cannot reduce the sizes of all buckets at the same time – removing a rule from one bucket means adding it to another. Allowing for this fact, the excess edge competition example shown in Section 1 is essentially caused by the uneven distribution of rules among different buckets $S(\cdot)$. Accordingly, our optimization objective should be a more even distribution of rules among buckets.

In the following, we formally define a metric to model the evenness of rule distribution over buckets. Given a binary SCFG G' and a binary SCFG rule $v \in G'$, $Q(v)$ is defined as the *cost function* that maps v to the size of the bucket $S(v, G')$:

$$Q(v) = |S(v, G')| \tag{1}$$

Obviously, all the binary rules in $S(v, G')$ share a common cost value $|S(v, G')|$. For example (Figure 3), both v_{13} and v_{22} are put into the same bucket $S("NP \text{ 会}", G')$, so $Q(v_{13}) = Q(v_{22}) = 2$.

The cost of the SCFG G' is computed by summing up all the costs of SCFG rules in it:

$$Q(G') = \sum\nolimits_{v \in G'} Q(v) \tag{2}$$

Back to our task, we are to find an equivalent binary SCFG G' of G with the lowest cost in terms of the cost function $Q(.)$ given in Equation (2):

$$G^* = \text{argmin}_{G' \in \mathcal{B}(G)} Q(G') \tag{3}$$

Next we will show how G^* is related to the evenness of rule distribution among different buckets. Let $S(G') = \{S_1, \ldots, S_M\}$ be the set of rule buckets containing rules in G', then the value of $Q(G')$ can also be written as:

$$Q(G') = \sum\nolimits_{1 \le i \le M} |S_i|^2 \tag{4}$$

Assume $Y_i = |S_i|$ is an empirical distribution of a discrete random variable Y, then the square deviation of the empirical distribution is:

$$\sigma^2 = \frac{1}{M} \sum\nolimits_i (|S_i| - \bar{Y})^2 \tag{5}$$

Noticing that $\Sigma|S_i| = |G'|$ and $\bar{Y} = |G'|/M$, Equation (5) can be written as:

$$\sigma^2 = \frac{1}{M} \left(Q(G') - \frac{|G'|^2}{M} \right) \tag{6}$$

Since both M and $|G'|$ are constants, minimizing the cost function $Q(G')$ is equivalent to minimizing the square deviation of the distribution of rules among different buckets. A binary SCFG with the lower cost indicates the rules are more evenly distributed in terms of derivation patterns on the source language side.

3.3 Static Cost Reduction

Before moving on discussing the algorithm which can optimize Equation (3) based on rule costs specified in Equation (1), we first present an algorithm to find the optimal solution to Equation (3) if we have known the cost setting of G^* and can use the costs as static values during binarization. Using this simplification, the problem of finding the binary SCFG G^* with minimal costs can be reduced to find the optimal binarization $\mathcal{B}^*(R_i)$ for each rule R_i in G.

To obtain $\mathcal{B}^*(R_i)$, we can employ a CKY-style binarization algorithm which builds a compact binarization forest for the rule R_i in bottom-up direction. The algorithm combines two adjacent spans of α_i each time, in which two spans can be combined if and only if they observe the BTG constraints – their translations are either sequentially or reversely adjacent in β_i, the target-side derivation of R_i. The key idea of this algorithm is that we only use the binarization tree with the lowest cost of each span for later combination, which can avoid enumerating all the possible binarization trees of R_i using dynamic programming.

Let α_p^q be the sub-sequence spanning from p to q on the source-side, $v[p, q]$ be optimal binarization tree spanning α_p^q, $Q_v[p, q]$ be the cost of $v[p, q]$, and $Q_r[p, q]$ be the cost of any binary rules whose source-side is α_p^q, then the cost of optimal binarization tree spanning α_p^q can be computed as:

$$Q_v[p, q] = \min_{p \le k \le q-1} (Q_r[p, q] + Q_v[p, k] + Q_v[k+1, q])$$

The algorithm is shown as follows:

CYK-based binarization algorithm

Input: a SCFG rule R_i and the cost function $Q(.)$.
Output: the lowest cost binarization on R_i

1: **Function** CKYBINARIZATION(R_i, Q)
2: **for** l = 2 to n **do** ▷ Length of span
3: **for** p = 1 to $n - l + 1$ **do** ▷ Start of span
4: $q = p + l$ ▷ End of span
5: **for** $k = p$ to $q - 1$ **do** ▷ Partition of span
6: **if** not CONSECUTIVE($T(p,k)$, $T(k+1,q)$) **then next loop**
7: $Q_r[p,q] \leftarrow Q(\alpha_p^q)$
8: $curCost \leftarrow Q_r[p,q]+Q_v[p,k]+Q_v[k+1,q]$
9: **if** $curCost < minCost$ **then**
10: $minCost \leftarrow curCost$
11: $v[p,q] \leftarrow$ COMBINE($v[p,k]$, $v[k+1,q]$)
12: $Q_v[p,q] \leftarrow minCost$
13: **return** $v[1,n]$
14: **Function** CONSECUTIVE((*a*, *b*), (*c*, *d*))
15: **return** ($b = c - 1$) **or** ($d = a - 1$)

where n is the number of tokens (consecutive terminals are viewed as a single token) on the source-side of R_i. COMBINE($v[p,k]$, $v[k+1,q]$) combines the two binary sub-trees into a larger sub-tree over α_p^q. $T(p,q) = (a,b)$ means that the non-terminals covering α_p^q have the consecutive indices ranging from a to b on the target-side. If the target non-terminal indices are not consecutive, we set $T(p,q) = (-1,-1)$. $Q(\alpha_p^q) = Q(v')$ where v' is any rule in the bucket $S(\alpha_p^q, G')$.

In the algorithm, lines 9-11 implement dynamic programming, and the function CONSECUTIVE checks whether the two spans can be combined.

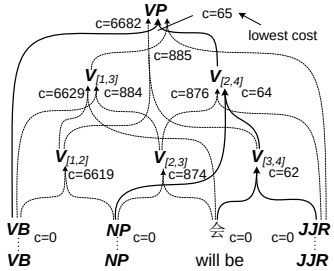


Figure 4: Binarization forest for an SCFG rule

$L(v)$	$Q(v)$	$L(v)$	$Q(v)$
VB NP	6619	VB NP 会	10
NP 会	874	NP 会 JJR	2
会 JJR	62	VB NP 会 JJR	1

Table 1: Sub-sequences and corresponding costs

Figure 4 shows an example of the compact forest the algorithm builds, where the solid lines indicate the optimal binarization of the rule, while other alternatives pruned by dynamic programming are shown in dashed lines. The costs for binarization trees are computed based on the cost table given in Table 1.

The time complexity of the CKY-based binarization algorithm is $\Theta(n^3)$, which is higher than that of the linear binarization such as the synchronous binarization (Zhang et al., 2006). But it is still efficient enough in practice, as there are generally only a few tokens ($n < 5$) on the source-sides of SCFG rules. In our experiments, the linear binarization method is just 2 times faster than the CKY-based binarization.

3.4 Iterative Cost Reduction

However, $Q(\cdot)$ cannot be easily predetermined in a static way as is assumed in Section 3.3 because it depends on G' and should be updated whenever a rule in G is binarized differently. In our work this problem is solved using the *iterative cost reduction* algorithm, in which the update of G' and the cost function $Q(\cdot)$ are coupled together.

Iterative cost reduction algorithm

Input: An SCFG G
Output: An equivalent binary SCFG G' of G

1: **Function** ITERATIVECOSTREDUCTION(G)
2: $G' \leftarrow G_0$
3: **for** each $v \in G_0$ **do**
4: $Q(v) = |S(v, G_0)|$
5: **while** $Q(G')$ does not converge **do**
6: **for** each $R_i \in G$ **do**
7: $G_{[-R_i]} \leftarrow G' - \mathcal{B}(R_i)$
8: **for** each $v \in \mathcal{B}(R_i)$ **do**
9: **for** each $v' \in S(v, G')$ **do**
10: $Q(v') \leftarrow Q(v') - 1$
11: $\mathcal{B}(R_i) \leftarrow$ CKYBINARIZATION(R_i, Q)
12: $G' \leftarrow G_{[-R_i]} \cup \mathcal{B}(R_i)$
13: **for** each $v \in \mathcal{B}(R_i)$ **do**
14: **for** each $v' \in S(v, G')$ **do**
15: $Q(v') \leftarrow Q(v') + 1$
16: **return** G'

In the *iterative cost reduction* algorithm, we first obtain an initial binary SCFG G_0 using the synchronous binarization method proposed in (Zhang et al., 2006). Then G_0 is assigned to an iterative variable G'. The cost of each binary rule in G_0 is computed based on G_0 according to Equation (1) (lines 3-4 in the algorithm).

After initialization, G' is updated by iteratively finding better binarization for each rule in G. The basic idea is: for each R_i in G, we remove the current binarization result for R_i from G' (line 7), while the cost function $Q(\cdot)$ is updated accordingly since the removal of binary rule $v \in \mathcal{B}(R_i)$ results in the reduction of the size of the corresponding bucket $S(v, G')$. Lines 8-10 im-

plement the cost reduction of each binary rule in the bucket $S(v, G')$.

Next, we find the lowest cost binarization for R_i based on the updated cost function $Q(\cdot)$ with the CKY-based binarization algorithm presented in Section 3.3 (line 11).

At last, the new binarization for R_i is added back to G' and $Q(\cdot)$ is re-updated to synchronize with this change (lines 12-15). Figure 5 illustrates the differences between the static cost reduction and the iterative cost reduction.

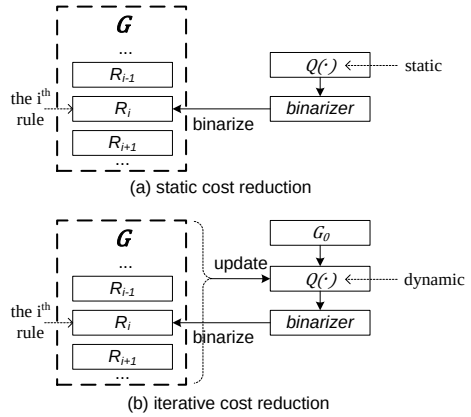


Figure 5: Comparison between the static cost reduction and the iterative cost reduction

The algorithm stops when $Q(G')$ does not decrease any more. Next we will show that $Q(G')$ is guaranteed not to increase in the iterative process.

For any $\mathcal{B}(R_i)$ on R_i, we have

$$Q\left(G_{[-R_i]} \cup \mathcal{B}(R_i)\right) = 2 \times Q\left(\mathcal{B}(R_i)\right) + |\mathcal{B}(R_i)| + Q\left(G_{[-R_i]}\right)$$

As both $|\mathcal{B}(R_i)|$ and $Q\left(G_{[-R_i]}\right)$ are constants with respect to $Q(\mathcal{B}(R_i))$, $Q\left(G_{[-R_i]} \cup \mathcal{B}(R_i)\right)$ is a linear function of $Q(\mathcal{B}(R_i))$, and the corresponding slope is positive. Thus $Q\left(G_{[-R_i]} \cup \mathcal{B}(R_i)\right)$ reaches the lowest value only when $Q(\mathcal{B}(R_i))$ reaches the lowest value. So $Q\left(G_{[-R_i]} \cup \mathcal{B}(R_i)\right)$ achieves the lowest cost when we replace the current binarization with the new binarization $\mathcal{B}^*(R_i)$ (line 12). Therefore $Q\left(G_{[-R_i]} \cup \mathcal{B}(R_i)\right)$ does not increase in the processing on each R_i (lines 7-15), and $Q(G')$ will finally converge to a local minimum when the algorithm stops.

4 Experiments

The experiments are conducted on Chinese-to-English translation in a state-of-the-art string-to-tree SMT system. All the results are reported in terms of case-insensitive BLEU4(%).

4.1 Experimental Setup

Our bilingual training corpus consists of about 350K bilingual sentences (9M Chinese words + 10M English words)[2]. Giza++ is employed to perform word alignment on the bilingual sentences. The parse trees on the English side are generated using the Berkeley Parser[3]. A 5-gram language model is trained on the English part of LDC bilingual training data and the Xinhua part of Gigaword corpus. Our development data set comes from NIST2003 evaluation data in which the sentences of more than 20 words are excluded to speed up the Minimum Error Rate Training (MERT). The test data sets are the NIST evaluation sets of 2005 and 2008.

Our string-to-tree SMT system is built based on the work of (Galley et al., 2006; Marcu et al., 2006), where both the minimal GHKM and SPMT rules are extracted from the training corpus, and the composed rules are generated by combining two or three minimal GHKM and SPMT rules. Before the rule extraction, we also binarize the parse trees on the English side using Wang et al. (2007) 's method to increase the coverage of GHKM and SPMT rules. There are totally 4.26M rules after the low frequency rules are filtered out. The pruning strategy is similar to the cube pruning described in (Chiang, 2007). To achieve acceptable translation speed, the beam size is set to 50 by default. The baseline system is based on the synchronous binarization (Zhang et al., 2006).

4.2 Binarization Schemes

Besides the baseline (Zhang et al., 2006) and iterative cost reduction binarization methods, we also perform right-heavy and random synchronous binarizations for comparison. In this paper, the random synchronous binarization is obtained by: 1) performing the CKY binarization to build the binarization forest for an SCFG rule; then 2) performing a top-down traversal of the forest. In the traversal, we randomly pick a feasible binarization for each span, and then go on the traversal in the two branches of the picked binarization.

Table 2 shows the costs of resulting binary SCFGs generated using different binarization methods. The costs of the baseline (left-heavy)

[2] LDC2003E14, LDC2003E07, LDC2005T06 and LDC2005T10

[3] http://code.google.com/p/berkeleyparser/

and right-heavy binarization are similar, while the cost of the random synchronous binarization is lower than that of the baseline method[4]. As expected, the iterative cost reduction method obtains the lowest cost, which is much lower than that of the other three methods.

Method	cost of binary SCFG G'
Baseline	4,897M
Right-heavy	5,182M
Random	3,479M
Iterative cost reduction	185M

Table 2: Costs of the binary SCFGs generated using different binarization methods.

4.3 Evaluation of Translations

Table 3 shows the performance of SMT systems based on different binarization methods. The iterative cost reduction binarization method achieves the best performance on the test sets as well as the development set. Compared with the baseline method, it obtains gains of 0.82 and 0.84 BLEU scores on NIST05 and NIST08 test sets respectively. Using the statistical significance test described by Koehn (2004), the improvements are significant ($p < 0.05$).

Method	Dev	NIST05	NIST08
Baseline	40.02	37.90	27.53
Right-heavy	40.05	37.87	27.40
Random	40.10	37.99	27.58
Iterative cost reduction	40.97*	38.72*	28.37*

Table 3: Performance (BLUE4(%)) of different binarization methods. * = significantly better than baseline ($p < 0.05$).

The baseline method and the right-heavy binarization method achieve similar performance, while the random synchronous binarization method performs slightly better than the baseline method, which agrees with the fact of the cost reduction shown in Table 2. A possible reason that the random synchronous binarization method can outperform the baseline method lies in that compared with binarizing SCFG in a fixed way, the random synchronous binarization tends to give a more even distribution of rules among buckets, which alleviates the problem of edge competition. However, since the high-frequency source sub-sequences still have high probabilities to be generated in the binarization and lead to the excess competing edges, it just achieves a very small improvement.

4.4 Translation Accuracy vs. Cost of Binary SCFG

We also study the impacts of cost reduction on translation accuracy over iterations in iterative cost reduction. Figure 6 and Figure 7 show the results on NIST05 and NIST08 test sets. We can see that the cost of the resulting binary SCFG drops greatly as the iteration count increases, especially in the first iteration, and the BLEU scores increase as the cost decreases.

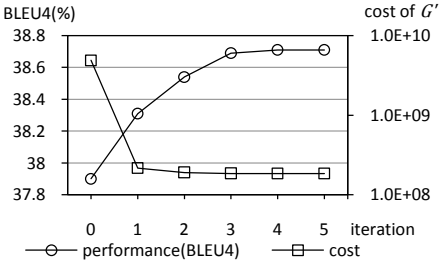


Figure 6: Cost of binary SCFG vs. BLEU4 (NIST05)

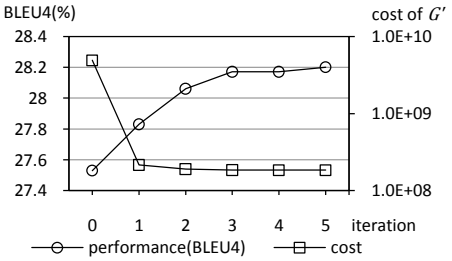


Figure 7: Cost of binary SCFG vs. BLEU4 (NIST08)

4.5 Impact of Beam Size

In this section, we study the impacts of beam sizes on translation accuracy as well as competing edges. To explicitly investigate the issue under large beam sizes, we use a subset of NIST05 and NIST08 test sets for test, which has 50 Chinese sentences of no longer than 10 words.

Figure 8 shows that the iterative cost reduction method is consistently better than the baseline method under various beam settings. Besides the experiment on the test set of short sentences, we also conduct the experiment on NIST05 test set. To achieve acceptable decoding speed, we range the beam size from 10 to 70. As shown in Figure 9, the iterative cost reduction method also outperforms the baseline method under various beam settings on the large test set.

Though enlarging beam size can reduce the search errors and improve the system performance, the decoding speed of string-to-tree SMT drops dramatically when we enlarge the beam size. The problem is more serious when long

[4] We perform random synchronous binarization for 5 times and report the average cost.

sentences are translated. For example, when the beam size is set to a larger number (e.g. 200), our decoder takes nearly one hour to translate a sentence whose length is about 20 on a 3GHz CPU. Decoding on the entire NIST05 and NIST08 test sets with large beam sizes is impractical.

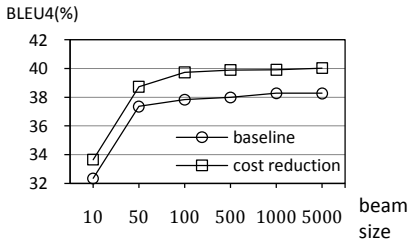


Figure 8: BLEU4 against beam size (small test set)

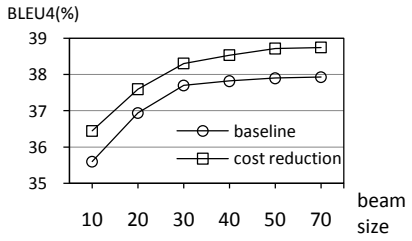


Figure 9: BLEU4 against beam size (NIST05)

Figure 10 compares the baseline method and the iterative cost reduction method in terms of translation accuracy against the number of edges proposed during decoding. Actually, the number of edges proposed during decoding can be regarded as a measure of the size of search space. We can see that the iterative cost reduction method outperforms the baseline method under various search effort.

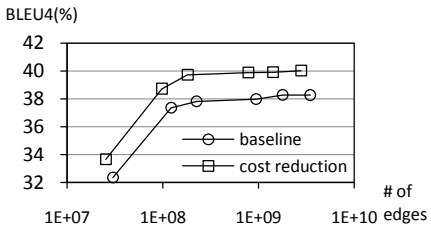


Figure 10: BLEU4 against competing edges

The experimental results of this section show that compared with the baseline method, the iterative cost reduction method can lead to much fewer edges (about 25% reduction) as well as the higher BLEU scores under various beam settings.

4.6 Edge Competition vs. Cost of Binary SCFG

In this section, we study the impacts of cost reduction on the edge competition in the chart cells of our CKY-based decoder. Two metrics are used to evaluate the degree of edge competition. They are the variance and the mean of the number of competing edges in the chart cells, where high variance means that in some chart cells the rules have high risk to be pruned due to the large number of competing edges. The same situation holds for the mean as well. Both of the two metrics are calculated on NIST05 test set, varying with the span length of chart cell.

Figure 11 shows the cost of resulting binary SCFG and the variance of competing edges against iteration count in iterative cost reduction. We can see that both the cost and the variance reduce greatly as the iteration count increases. Figure 12 shows the case for mean, where the reduction of cost also leads to the reduction of the mean value. The results shown in Figure 11 and Figure 12 indicate that the cost reduction is helpful to reduce edge competition in the chart cells.

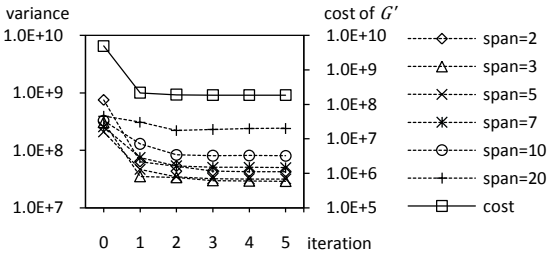


Figure 11: Cost of binary SCFG vs. variance of competing edge number (NIST05)

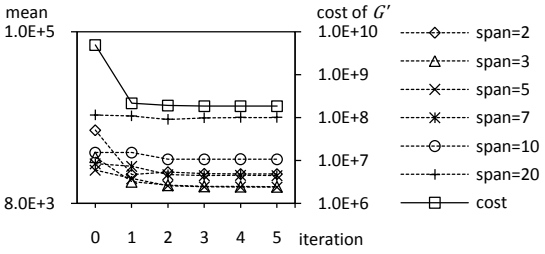


Figure 12: Cost of binary SCFG vs. mean of competing edge number (NIST05)

We also perform decoding without pruning (i.e. beam size = ∞) on a very small set which has 20 sentences of no longer than 7 words. In this experiment, the baseline system and our iterative cost reduction based system propose 14,454M and 10,846M competing edges respectively. These numbers can be seen as the real numbers of the edges proposed during decoding instead of an approximate number observed in the pruned search space. It suggests that our method can reduce the number of the edges in real search space effectively. A possible reason to

this result is that the cost reduction based binarization could reduce the probability of rule mismatching caused by binarization, which results in the reduction of the number of edges proposed during decoding.

5 Conclusion and Future Work

This paper introduces a new binarization method, aiming at choosing better binarization for SCFG-based SMT systems. We demonstrate the effectiveness of our method on a state-of-the-art string-to-tree SMT system. Experimental results show that our method can significantly outperform the conventional synchronous binarization method, which indicates that better binarization selection is very beneficial to SCFG-based SMT systems.

In this paper the cost of a binary rule is defined based on the competition among the binary rules that have the same source-sides. However, some binary rules with different source-sides may also have competitions in a chart cell. We think that the cost of a binary rule can be better estimated by taking the rules with different source-sides into account. We intend to study this issue in our future work.

Acknowledgements

The authors would like to thank the anonymous reviewers for their pertinent comments, and Xinying Song, Nan Duan and Shasha Li for their valuable suggestions for improving this paper.

References

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. 2006. Multilevel Coarse-to-Fine PCFG Parsing. In *Proc. of HLT-NAACL* 2006, New York, USA, 168-175.

Eugene Charniak, Sharon Goldwater, and Mark Johnson. 1998. Edge-Based Best-First Chart Parsing. In *Proc. of the Six Workshop on Very Large Corpora*, pages: 127-133.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proc. of ACL 2005*, Ann Arbor, Michigan, pages: 263-270.

David Chiang. 2007. Hierarchical Phrase-based Translation. *Computational Linguistics*. 33(2): 202-208.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable Inference and Training of Context-Rich Syntactic Translation Models. In *Proc. of ACL 2006*, Sydney, Australia, pages: 961-968.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. of HLT-NAACL 2004*, Boston, USA, pages: 273-280.

Liang Huang. 2007. Binarization, Synchronous Binarization, and Target-side binarization. In *Proc. of HLT-NAACL 2007 / AMTA workshop on Syntax and Structure in Statistical Translation*, New York, USA, pages: 33-40.

Tadao Kasami. 1965. An Efficient Recognition and Syntax Analysis Algorithm for Context-Free Languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, Massachusetts.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proc. of EMNLP 2004*, Barcelona, Spain , pages: 388–395.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. of EMNLP 2006*, Sydney, Australia, pages: 44-52.

Giorgio Satta and Enoch Peserico. 2005. Some Computational Complexity Results for Synchronous Context-Free Grammars. In *Proc. of HLT-EMNLP 2005*, Vancouver, pages: 803-810.

L. Shapiro and A. B. Stephens. 1991. Bootstrap percolation, the Schr̈oder numbers, and the *n*-kings problem. SIAM Journal on Discrete Mathematics, 4(2):275-280.

Xinying Song, Shilin Ding and Chin-Yew Lin. 2008. Better Binarization for the CKY Parsing. In *Proc. of EMNLP 2008*, Hawaii, pages: 167-176.

Yoshimasa Tsuruoka and Junichi Tsujii. 2004. Iterative CKY Parsing for Probabilistic Context-Free Grammars. In *Proc. of IJCNLP 2004*, pages: 52-60.

Wei Wang and Kevin Knight and Daniel Marcu. 2007. Binarizing Syntax Trees to Improve Syntax-Based Machine Translation Accuracy. In *Proc. of EMNLP-CoNLL 2007*, Prague, Czech Republic, pages: 746-754.

D. H. Younger. 1967. Recognition and Parsing of Context-Free Languages in Time n^3. *Information and Control*, 10(2):189-208.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous Binarization for Machine Translation. In *Proc. of HLT-NAACL 2006*, New York, USA, pages: 256- 263.

Accuracy-Based Scoring for DOT: Towards Direct Error Minimization for Data-Oriented Translation

Daniel Galron
CIMS
New York University
galron@cs.nyu.edu

Sergio Penkale, Andy Way
CNGL
Dublin City University
{spenkale,away}
@computing.dcu.ie

I. Dan Melamed
AT&T Shannon Laboratory
{lastname}
@research.att.com

Abstract

In this work we present a novel technique to rescore fragments in the Data-Oriented Translation model based on their contribution to translation accuracy. We describe three new rescoring methods, and present the initial results of a pilot experiment on a small subset of the Europarl corpus. This work is a proof-of-concept, and is the first step in directly optimizing translation decisions solely on the hypothesized accuracy of potential translations resulting from those decisions.

1 Introduction

The Data-Oriented Translation (DOT) (Poutsma, 2000) model is a tree-structured translation model, in which linked subtree fragments extracted from a parsed bitext are composed to cover a source-language sentence to be translated. Each linked fragment pair consists of a source-language side and a target-language side, similar to (Wu, 1997). Translating a new sentence involves composing the linked fragments into derivations so that a new source-language sentence is covered by the source tree fragments of the linked pairs, where the yields of the target-side derivations are the candidate translations. Derivations are scored according to their likelihood, and the translation is selected from the derivation pair with the highest score. However, we have no reason to believe that maximizing likelihood is the best way to maximize translation accuracy – likelihood and accuracy do not necessarily correlate well.

We can frame the problem as a search problem, where we are searching a space of derivations for the one that yields the highest scoring translation. By putting weights on the derivations in the search space, we wish to point the decoder in the direction of the optimal translation. Since we want the decoder to find the translation with the highest evaluation score, we would want to score the derivations with weights that correlate well with the particular evaluation measure in mind.

Much of the work in the MT literature has focused on the scoring of translation decisions made. (Yamada and Knight, 2001) follow (Brown et al., 1993) in using the noisy channel model, by decomposing the translation decisions modeled by the translation model into different types, and inducing probability distributions via maximum likelihood estimation over each decision type. This model is then decoded as described in (Yamada and Knight, 2002). This type of approach is also followed in (Galley et al., 2006).

There has been some previous work on accuracy-driven training techniques for SMT, such as MERT (Och, 2003) and the Simplex Armijo Downhill method (Zhao and Chen, 2009), which tune the parameters in a linear combination of various phrase scores according to a held-out tuning set. While this does tune the relative weights of the scores to maximize the accuracy of candidates in the tuning set, the scores themselves in the linear combination are not necessarily correlated with the accuracy of the translation. Tillmann and Zhang (2006) present a procedure to directly optimize the global scoring function used by a phrase-based decoder on the accuracy of the translations. Similarly to MERT, Tillmann and Zhang estimate the parameters of a weight vector on a linear combination of (binary) features using a global objective function correlated with BLEU (Papineni et al., 2002).

In this work, we prototype some methods for moving directly towards incorporating a measure of the translation quality of each fragment used, bringing DOT more into the mainstream of current SMT research. In Section 2 we describe probability-based DOT fragment scoring. In Section 3 we describe our rescoring setup and the

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 371–380,
Singapore, 6-7 August 2009.

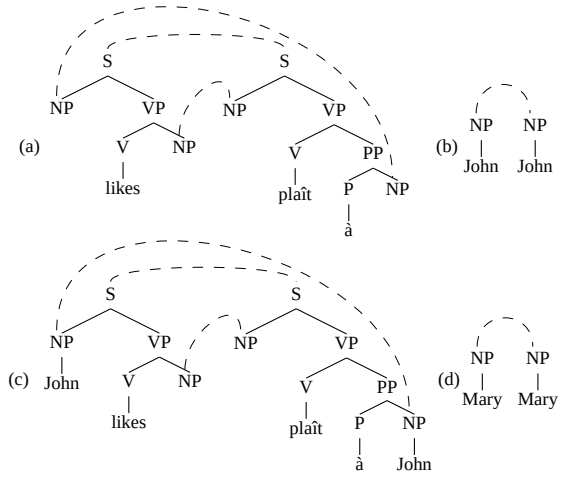


Figure 1: Example DOT Fragments.

three rescoring methods. In Section 4, we describe our experiments. In Section 5 we compare the results of rescoring the fragments with the three methods. In Section 6 we discuss some of the decisions that are affected by our rescoring methods. Finally, we discuss the next steps in training the DOT system by optimizing over a translation accuracy-based objective function in Section 7.

2 DOT Scoring

As described in previous work (Poutsma, 2000; Hearne and Way, 2003), DOT scores translations according to the probabilities of the derivations, which are in turn computed from the relative frequencies of linked tree fragments in a parallel treebank. Linked fragment pairs are conditionally independent, so the score of a derivation is the product of the probabilities of all the linked fragments used. To find the probability of a translation, DOT marginalizes over the scores of all derivations yielding the translation.

From a parallel treebank aligned at the subsentential level, we extract all possible linked fragment pairs by first selecting all linked pairs of nodes in the treebank to be the roots of a new subtree pair, and then selecting a (possibly empty) set of linked node pairs that are descendants of the newly selected fragment roots and deleting all subtree pairs dominated by these nodes. Leaves of fragments can either be terminals, or non-terminal *frontier nodes* where we can compose other fragments (c.f. (Eisner, 2003)). We give example DOT fragment pairs in Figure 1.

Given two subtree pairs $\langle s_1, t_1 \rangle$ and $\langle s_2, t_2 \rangle$, we can compose them using the DOT composition operator $\circ$ if the leftmost non-terminal frontier node of s_1 is equal to the root node of s_2, and the leftmost non-terminal frontier node of s_1's *linked counterpart* in t_1 is equal to the root node of t_2. The resulting tree pair consists of a copy of s_1 where s_2 has been inserted at the leftmost frontier node, and a copy of t_1 where t_2 has been inserted at the node linked to s_1's leftmost frontier node (Hearne and Way, 2003).

In Figure 1, fragment pair (a) is a fragment with two open substitution sites. If we compose this fragment pair with fragment pair (b), the source side composition must take place on the leftmost non-terminal frontier node (the leftmost NP). On the target side we compose on the frontier linked to the leftmost source side non-terminal frontier. The result is fragment pair (c). If we now compose the resulting fragment pair with fragment pair (d), we obtain a fragment pair with no open substitution sites whose source-side yield is *John likes Mary* and whose target-side yield is *Mary plaît à John*. Note that there are two different derivations using the fragment pairs in Figure 1 that result in the same fragment pair, namely (a) $\circ$ (b) $\circ$ (d), and (c) $\circ$ (d).

For a given linked fragment pair $\langle d_s, d_t \rangle$, the probability assigned to it is

$$P(\langle d_s, d_t \rangle) = \frac{|\langle d_s, d_t \rangle|}{\sum_{r(u_s)=r(d_s) \wedge r(u_t)=r(d_t)} |\langle u_s, u_t \rangle|} \quad (1)$$

where $|\langle d_s, d_t \rangle|$ is the number of times the fragment pair $\langle d_s, d_t \rangle$ is found in the bitext, and $r(d)$ is the root nonterminal of d. Essentially, the probability assigned to the fragment pair is the relative frequency of the fragment pair to the pair of nonterminals that root the fragments.

Then, with the assumption that DOT fragments are conditionally independent, the probability of a derivation is

$$\begin{aligned} P(\mathbf{d}) &= P(\langle d_s, d_t \rangle_1 \circ \ldots \circ \langle d_s, d_t \rangle_N) \\ &= \prod_i P(\langle d_s, d_t \rangle_i) \end{aligned} \quad (2)$$

In the original DOT formulation, DOT disambiguated translations according to their probabilities. Since a translation can have many possible derivations, to obtain the probability of a translation it is necessary to marginalize over the distinct derivations yielding a translation. The probability of a translation w_t of a source sentence w_s, is

given by (3):

$$P(w_s, w_t) = \sum_{\mathbf{d} \in \mathbf{D}} P(\mathbf{d}_{\langle w_s, w_t \rangle}) \quad (3)$$

and the translation is chosen so as to maximize (4):

$$\hat{w_t} = \underset{w_t}{\operatorname{argmax}} P(w_s, w_t) \quad (4)$$

Hearne and Way (2006) examined alternative disambiguation strategies. They found that rather than disambiguating on the translation probability, the translation quality would improve by disambiguating on the derivation probability, as in (5):

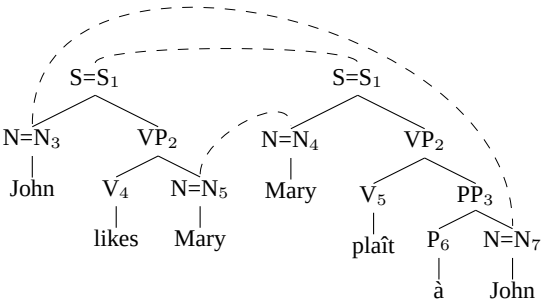
$$\hat{w_t} = \underset{\mathbf{d}}{\operatorname{argmax}} P(\mathbf{d}) \quad (5)$$

Our analysis suggest that this is because many derivations with very low probabilities generate the same, poor translation. When applying Equation (3) to marginalize over those derivations, the resulting score is higher for the poor translation than a better translation with fewer derivations but where the derivations had higher likelihood.

Using the DOT model directly is difficult – the number of fragments extracted from a parallel treebank is exponential in the size of the treebank. Therefore we use the Goodman reduction of DOT (Hearne, 2005) to create an isomorphic PCFG representation of the DOT model that is linear in the size of the treebank. The idea behind the Goodman reduction is that rather than storing fragments in the grammar and translating via composition, we simultaneously build up the fragments using the PCFG reduction and compose them together. To perform the reduction, we first relabel the two linked nodes (X, Y) with the new label X=Y. We then label each node in the parallel treebank with a unique Goodman index. Each binary-branching node and its two children can be internal or root/frontier. We add rules to the grammar reflecting the role that each node can take, keeping unaligned nodes as fragment-internal nodes. So in the case where a node and both of its children are aligned, we commit 8 rules into the grammar, as follows:

LHS → RHS1 RHS2	LHS+a → RHS1 RHS2
LHS → RHS1+b RHS2	LHS+a → RHS1+b RHS2
LHS → RHS1 RHS2+c	LHS+a → RHS1 RHS2+c
LHS → RHS1+b RHS+c	LHS+a → RHS1+b RHS2+c

A category label which ends in a '+' symbol followed by a Goodman index is fragment-internal and all other nodes are either fragment roots or



Source PCFG		Target PCFG	
S=S → N=N VP+2	0.5	S=S → N=N VP+2	0.5
S=S → N=N+3 VP+2	0.5	S=S → N=N+4 VP+2	0.5
S=S+1 → N=N VP+2	0.5	S=S+1 → N=N VP+2	0.5
S=S+1 → N=N+3 VP+2	0.5	S=S+1 → N=N+4 VP+2	0.5
N=N → John	0.5	N=N → Mary	0.5
N=N+3 → John	1	N=N+4 → Mary	1
VP+2 → V+4 N=N	0.5	VP+2 → V+5 PP+3	1
VP+2 → V+4 N=N+5	0.5	V+5 → plaît	1
V+4 → likes	1	PP+3 → P+6 N=N	0.5
N=N → Mary	0.5	PP+3 → P+6 N=N+7	0.5
N=N+5 → Mary	1	P+6 → à	1
		N=N → John	0.5
		N=N+7 → John	1

Figure 2: A parallel tree and its corresponding Goodman reduction.

frontier nodes. A fragment pair, then, is a pair of subtrees in which the root does not have an index, all internal nodes have indices, and all the leaves are either terminals or un-indexed nodes. We give an example Goodman reduction in Figure 2.

While we store the source grammar and the target grammar separately, we also keep track of the correspondence between source and target Goodman indices and can easily identify the alignments according to the Goodman indices. Probabilities for the PCFG rules are computed monolingually as in the standard Goodman reduction for DOP (Goodman, 1996). In decoding with the Goodman reduction, we first find the *n*-best parses on the source side, and for each source fragment, we construct the *k*-best fragments on the target side. We finally compute the bilingual derivation probabilities by multiplying the source and target derivation probabilities by the target fragment relative frequencies conditioned on the source fragment.

There are a few problems with a likelihood-based scoring scheme. First, it is not clear that if a fragment is more likely to be seen in training data then it is more likely to be used in a correct translation of an unseen sentence. In our analysis of the candidate translations of the DOT system, we observed that frequently, the highest-likelihood candidate translation output by the system was not the highest-accuracy candidate inferred. An additional problem is that, as described in (Johnson, 2002), the relative frequency estimator for DOP

(and by extension, DOT) is known to be biased and inconsistent.

3 Accuracy-Based Fragment Scoring

In our work, we wish to incorporate a measure of fragment accuracy into the scoring. To do so, we reformulate the scoring of DOT as log-linear rather than probabilistic, in order to incorporate non-likelihood features into the derivation scores. For all tree fragment pairs $\langle d_s, d_t \rangle$, let

$$l(\langle d_s, d_t \rangle) = \log(p(\langle d_s, d_t \rangle)) \quad (6)$$

The general form of a rescored tree fragment will be

$$s(\langle d_s, d_t \rangle) = \alpha_0 l(\langle d_s, d_t \rangle) + \sum_{i=1}^{k} \alpha_i f_i(\langle d_s, d_t \rangle) \quad (7)$$

where each α_i is the weight of that term in the final score, and each $f_i(d)$ is a feature. In this work, we only consider $f_1(d)$, an accuracy-based score, although in future work we will consider a wide variety of features in the scoring function, including combinations of the different scoring schemes described below, binary lexical features, binary source-side syntactic features, and local target side features. The score of a derivation is now given by (8):

$$\begin{aligned} s(d) &= s(\langle d_s, d_t \rangle_1 \circ \ldots \circ \langle d_s, d_t \rangle_N) \\ &= \sum_i s(\langle d_s, d_t \rangle_i) \end{aligned} \quad (8)$$

In order to disambiguate between candidate translations, we follow (Hearne and Way, 2006) by using Equation (5).

3.1 Structured Fragment Rescoring

In all our approaches, we rescore fragments according to their contribution to the accuracy of a translation. We would like to give fragments that contribute to good translations relatively high scores, and give fragments that contribute to bad translations relatively low scores, so that during decoding fragments that are known to contribute to good translations would be chosen over those that are known to contribute to bad translations. Furthermore, we would like to score each fragment in a derivation independently, since bad translations may contain good fragments, and vice-versa.

In practice, it is infeasible to rescore only those fragments seen during the rescoring process, due to the Goodman reduction for DOT. If we were to properly rescore each fragment, a new rule would need to be added to the grammar for each rule appearing in the fragment. Since the number of fragments is exponential, this would lead to a substantial increase in grammar size. Instead, we rescore the individual rules in the fragments, by evenly dividing the total amount of scoring mass among the rules of the particular fragment, and then assigning them the average of the rule scores over all fragments in which they appear. That is for each rule r in a fragment f consisting of $c_f(r)$ rules with score $\delta(f)$, the score of the rule is given as:

$$s(r) = \frac{\sum_{f:r \in f} \delta(f)/c_f(r)}{|f|} \quad (11)$$

This has the further advantage that we are allowing fragments that were unseen during tuning to be rescored according to previously seen fragment substructures.

To implement this scheme, we select a set of oracle translations for each sentence in the tuning data by evaluating all the candidate translations against the gold standard translation using the F-score (Turian et al., 2003), and selecting those with the highest F_1-measure, with exponent 1. We use GTM, rather than BLEU, because BLEU is not known to work well on a per-sentence level (Lavie et al., 2004) as needed for oracle selection. We then compare all the *target-side* fragments inferred in the translation process for each candidate translation against the fragments that yielded the oracles. There are two relevant parts of the fragments – the internal yields (i.e. the terminal leaves of the fragment) and the substitution sites (i.e. the frontiers where other fragments attach). We score the fragments rooted at the substitution sites separately from the parent fragment. We can uniquely identify the set of fragments that can be rooted at substitution sites by determining the span of the linked source-side derivation.

To compare two fragments, we define an edit distance between them. For a given fragment d, let $r(d)$ be the root of the fragment, let $r(d) \rightarrow rhs1$ be the left subtree of $r(d)$, and let $r(d) \rightarrow rhs2$ be the right subtree. The difference between a candidate fragment d_c and an oracle fragment d_{gs} is given by the equations in Table 1.

These equations define a minimum edit distance between two fragment trees, allowing subfragment order inversion, insertion, and deletion

$$\delta(d_c, d_{gs}) = \begin{cases} 0 & \text{if } d_c = d_{gs} \\ 1 & \text{if } d_c \neq d_{gs} \end{cases} \text{ Base case: } d_c \text{ and } d_{gs} \text{ are unary subtrees or substitution sites} \quad (9)$$

$$\delta(d_c, d_{gs}) = \min \begin{cases} \delta(d_c \rightarrow rhs1, d_{gs} \rightarrow rhs1) + \delta(d_c \rightarrow rhs2, d_{gs} \rightarrow rhs2), \\ \delta(d_c \rightarrow rhs2, d_{gs} \rightarrow rhs1) + \delta(d_c \rightarrow rhs1, d_{gs} \rightarrow rhs2) + 1, \\ \delta(d_c, d_{gs} \rightarrow rhs1) + |y(d_{gs} \rightarrow rhs2)|, \\ \delta(d_c, d_{gs} \rightarrow rhs2) + |y(d_{gs} \rightarrow rhs1)|, \\ \delta(d_c \rightarrow rhs1, d_{gs}) + |y(d_c \rightarrow rhs2)|, \\ \delta(d_c \rightarrow rhs2, d_{gs}) + |y(d_c \rightarrow rhs1)| \end{cases} \quad (10)$$

Table 1: The recursive relation defining the fragment difference between two fragments.

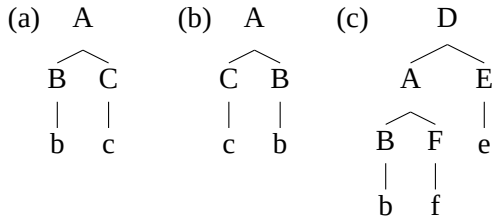


Figure 3: Comparing trees (a) and (b) with our distance metric yields a value of 1. The difference between trees (a) and (c) is 2, and for trees (b) and (c) the distance is 3.

as edit operations. For example, the only difference between trees (a) and (b) in Figure 3 is that their children have been inverted. To compare these trees using our distance metric, we first compute the first argument of the $\min$ function in Equation (10), directly comparing the structure of each immediate subtree. We then compute the second argument, obtaining the cost of performing an inversion, and finally compute the remaining arguments, assessing the cost of allowing each tree to be a direct subtree of the other. The result of this computation is 1, representing the inversion operation required to transform tree (a) into tree (b). If we compare trees (a) and (c) in Figure 3, we obtain a value of 2, given that the minimum operations required to transform tree (a) into tree (c) are inserting an additional subtree at the top level and then substituting the subtree rooted by C for the subtree rooted by F. If we compare tree (b) with tree (c) then the distance is 3, since we are now required to also replace the subtree rooted by C by the one rooted by B.

Since it is not efficient to compute the differences directly, we utilize common substructures and derive a dynamic programming implementation of the recursion. We compare each fragment against the set of oracle fragments for the same source span, and select the lowest cost as the score, assigning the candidate the negative difference between it and the oracle fragment it is most similar to, as in (12):

$$f(\langle d_s, d_t \rangle) = \max_{\langle d_s^o, d_t^o \rangle \in \mathbf{D}^o : d_s^o = d_s} -\delta(d_t, d_t^o) \quad (12)$$

In practice, given the Goodman reduction for DOT, we divide the fragment score by the number of rules in the fragment, and assign the average of those scores for each rule instance across all fragments rescored.

3.2 Normalized Structured Fragment Rescoring

In the structured fragment rescoring scheme, the scores that the fragments are assigned are the unnormalized edit distances between the two fragments. It may be better to normalize the fragment scores, rather than using the minimum number of tree transformations to convert one fragment into the other. We would expect that when comparing larger fragments, on average there would be more transformations needed to change one into the other than when comparing small fragments. However in the previous scheme, small fragments would have higher scores than large fragments, since fewer differences would be observed. The normalized score is given in (13):

$$f(\langle d_s, d_t \rangle) = \max_{\langle d_s^o, d_t^o \rangle \in \mathbf{D}^o : d_s^o = d_s} \log(1 - \delta(d_t, d_t^o)/ \max(|d_t|, |d_t^o|)) \quad (13)$$

Essentially, we are normalizing the edit distance by the maximum edit distance possible, namely the size of the largest fragment of the two being compared.

3.3 Fragment Surface Rescoring

The disadvantage of the minimum tree fragment edit approach is that it explicitly takes the internal

syntactic structure of the fragment into account. In comparing two fragments, they may have the same (or very similar) surface yields, but different internal structures. The previous approach would penalize the candidate fragment, even if its yield is quite close to the oracle. In this rescoring method, we extract the leaves of the candidate and oracle fragments, representing the substitution sites by the source span which their fragments cover. We then compare them using the Damerau-Levenshtein distance $\delta_{dl}(d_c, d_{gs})$ (Damerau, 1964) between the two fragment yields, and score them as in (14):

$$f(\langle d_s, d_t \rangle) = \max_{\langle d_s^o, d_t^o \rangle \in \mathbf{D}^o : d_s^o = d_s} -\delta_{dl}(d_t, d_t^o) \quad (14)$$

In Equation (14) we are selecting the maximal score for $\langle d_s, d_t \rangle$ from its comparison to all the possible corresponding oracle fragments. In this way, we are choosing to score $\langle d_s, d_t \rangle$ against the oracle fragment it is closest to.

4 Experiments

For our pilot experiments, we tested all the rescoring methods in the previous section on Spanish-to-English translation against the relative-frequency baseline. We randomly selected 10,000 sentences from the Europarl corpus (Koehn, 2005), and parsed and aligned the bitext as described in (Tinsley et al., 2009). From the parallel treebank, we extracted a Goodman reduction DOT grammar, as described in (Hearne, 2005), although on an order of magnitude greater amount of training data. Unlike (Bod, 2007), we did not use the unsupervised version of DOT, and did not attempt to scale up our amount of training data to his levels, although in ongoing work we are optimizing our system to be able to handle that amount of training data. To perform the rescoring, we randomly chose an additional 30K sentence pairs from the Spanish-to-English bitext. We rescored the grammar by translating the source side of the 10K training sentence pairs and 10K of the additional sentences, and using the methods in Section 3 to score the fragments derived in the translation process. We then performed the same experiment translating the full 40K-sentence set. Rules in the grammar that were not used during tuning were rescored using a default score defined to be the median of all scores observed.

Our system performs translation by first obtaining the n-best parses for the source sentences and then computing the k-best bilingual derivations for each source parse. In our experiments we used beams of $n = 10,000$ and $k = 5$. We also experimented with different values of α_0 and α_1 in Equation (7). We set these parameters manually, although in future work we will automatically tune them, perhaps using a MERT-like algorithm.

We tested our rescored grammars on a set of 2,000 randomly chosen Europarl sentences, and used a set of 200 randomly chosen sentences as a development test set. [1]

	BLEU	NIST	F-SCORE
Baseline	8.78	3.582	38.21

		2-8	4-6	5-5	6-4	8-2
BLEU	SFR	10.30	10.31	**10.32**	10.27	10.08
	NSFR	8.31	9.37	9.53	9.66	9.90
	FSR	10.19	10.25	10.18	10.19	9.93
NIST	SFR	3.792	3.805	**3.808**	3.800	3.781
	NSFR	3.431	3.638	3.661	3.693	3.722
	FSR	3.784	3.799	3.792	3.795	3.764
F-SCORE	SFR	**40.92**	40.82	40.86	40.84	40.78
	NSFR	37.53	39.50	39.93	40.38	40.78
	FSR	40.83	40.85	40.87	40.91	40.67

Table 2: Results on test set. Rescoring on 20K sentences. *SFR* stands for Structured Fragment Rescoring, *NSFR* for Normalized SFR and *FSR* for Fragment Surface Rescoring. *system-i-j* represents the corresponding system with $\alpha_0 = i$ and $\alpha_1 = j$. Underlined results are statistically significantly better than the baseline at $p = 0.01$.

	BLEU	NIST	F-SCORE
Baseline	8.78	3.582	38.21

		2-8	4-6	5-5	6-4	8-2
BLEU	SFR	**10.59**	10.58	10.41	10.38	10.08
	NSFR	8.61	9.71	9.90	9.96	9.93
	FSR	10.49	10.48	10.35	10.38	10.06
NIST	SFR	**3.841**	3.835	3.810	3.807	3.785
	NSFR	3.515	3.694	3.713	3.734	3.727
	FSR	3.834	3.833	3.820	3.816	3.784
F-SCORE	SFR	**41.12**	40.99	40.86	40.88	40.75
	NSFR	38.16	40.39	40.69	40.90	40.75
	FSR	41.03	41.02	41.01	40.98	40.72

Table 3: Results on test set. Rescoring on 40K sentences. Underlined are statistically significantly better than the baseline at $p = 0.01$.

5 Results

Translation quality results can be found in Tables 2 and 3. In these tables, columns labeled i-j indicate that the corresponding system was trained using parameters $\alpha_0 = i$ and $\alpha_1 = j$ in Equation 7. Statistical significance tests for NIST and BLEU were performed using Bootstrap Resampling (Koehn, 2004).

[1] All sentences, including the ones used for training, were limited to a length of at most 20 words.

	BLEU		NIST		F-SCORE	
Baseline	10.82		3.493		42.31	
		2-8	4-6	5-5	6-4	8-2
BLEU	SFR	11.34	**12.12**	11.94	11.97	11.78
	NSFR	9.68	10.99	11.38	11.63	11.30
	FSR	11.40	11.49	11.72	11.91	11.72
NIST	SFR	3.653	**3.727**	3.723	3.708	3.694
	NSFR	3.376	3.530	3.554	3.616	3.572
	FSR	3.655	3.675	3.698	3.701	3.675
F-SCORE	SFR	44.84	**45.47**	45.36	45.33	45.08
	NSFR	41.44	43.38	44.18	44.79	44.26
	FSR	44.68	44.91	45.15	45.19	44.82

Table 4: Results on development test set. Rescoring on 40K sentences.

As Table 2 indicates, all three rescoring methods significantly outperform the relative frequency baseline. The unnormalized structured fragment rescoring method performed the best, with the largest improvement of 1.5 BLEU points, a 17.5% relative improvement. We note that the BLEU scores for both the baseline and the experiments are low. This is to be expected, because the grammar is extracted from a very small bitext especially when the heterogeneity of the Europarl corpus is considered. In our analysis, only 32.5 percent of the test sentences had a complete source-side parse, meaning that a lot of structural information is lost contributing to arbitrary target-side ordering. In these experiments we did not use an additional language model. DOT (and many other syntax-based SMT systems) essentially have the target language model encoded within the translation model, since the inferences derived during translations link source structures to target structures, so in principle, no additional language model should be necessary. Furthermore, we only evaluate against a single reference, which also contributes to the lowering of absolute scores. To provide a sanity check against a state-of-the-art system, we trained the Moses phrase-based MT system (Koehn et al., 2007) using our training corpus, using no language model and using uniform feature weights, to provide a fair comparison against our baseline. We used this system to decode our development test set, and as a result we obtained a BLEU score of 10.72, which is comparable to the score obtained by our baseline on the same set.

When we scale up to tuning on 40,000 sentences we see an improvement in BLEU scores as well, as shown in Table 3. When tuning on 40K sentences, we observe an increase of 1.81 BLEU points on the best-performing system, which is a 20.6% improvement over the baseline. We note that rescoring on 20K sentences rescores approximately 275,000 rules out of 655,000 in the grammar, whereas rescoring on 40K sentences rescores approximately 280,000.

To analyze the benefits of the rescored grammar, we set aside a separate development set that we decoded with the grammar trained on 40K sentences. The results are presented in Table 4. The analysis is presented in Section 6.

Interestingly, there is a large difference between the normalized and unnormalized versions of the SFR scoring scheme. Our analysis suggests that the differences are mostly due to numerical issues, namely the difference in magnitude between the NSFR scores and the likelihood scores in the linear combination, and the default value assigned when the NSFR score was zero. In ongoing work, we are working to address these issues.

For most configurations the difference between SFR and FSR was not statistically significant at $p = 0.05$. Our analysis indicated that surface differences tended to co-occur with structural differences. We hypothesize that as we scale up to larger and more ambiguous grammars, the system will infer more derivations with the same yields, rendering a larger difference between the quality of the two scoring mechanisms.

6 Discussion

To analyze the advantages and disadvantages of our approach over the baseline, we closely examined and compared the derivations made on the devset translation by the SFR-scored grammar and the likelihood-scored grammar. Although the BLEU scores are rather low, there were several sentences in which the SFR-scored grammar showed a marked improvement over the baseline. We observed two types of improvements.

The first is where the rescored grammar gave us translations that, while still generally bad, were closer to the gold standard than the baseline translation. For example, the Spanish sentence “Y en tercer lugar , está el problema de la aplicación uniforme del Derecho comunitario .” translates into the gold standard “Thirdly , we have the problem of the uniform application of Community law .” The baseline grammar translates the sentence as “on third place , Transport and Tourism . I are the problems of the implementation standardised is the EU law .” with a GTM F-Score of 0.378,

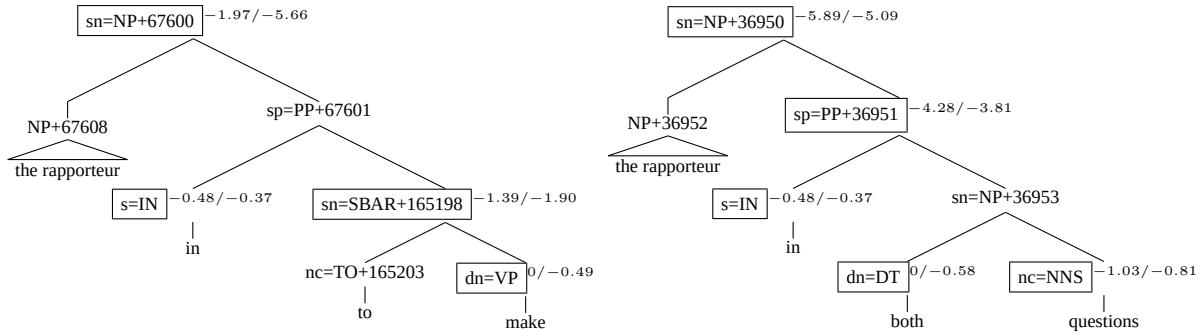


Figure 4: Target side of the highest-scoring translations for a sentence, according to the baseline system (left) and the SFR system (right). Boxed nodes are substitution sites. Scores in superscripts denote the score of the sub-derivation according to the baseline and to the SFR system.

and the rescored grammar outputs the translation "to there in the third place , I are the problem of the implementation standardised is the Community law .", with an F-Score of 0.5. While many of the fragments in the derivations that yielded these two translations differ, the ones we would like to focus on are the fragments that yield the translation of "comunitario". The grammar contains several competing unary fragment pairs for "comunitaro". In the baseline grammar, the pair (`aq=NNP` → `comunitario`, `aq=NNP` → `EU`) has a score of -0.693147, whereas the pair (`aq=NNP` → `comunitario`, `aq=NNP` → `Community`) has a score of -1.38629. In the rescored grammar however, (`aq=NNP` → `comunitario`, `aq=NNP` → `EU`) has a score of -0.762973, whereas (`aq=NNP` → `comunitario`, `aq=NNP` → `Community`) has a score of -0.74399. In effect, the rescoring scheme rescored the word alignment itself. This suggests that in future work, it may be possible to integrate a word aligner or fragment aligner directly into the MT training method.

The other improvement was where the baseline and the SFR-scored grammar output translations of roughly the same quality according to the evaluation measure, yet in terms of human evaluation, the SFR translation was much better than the baseline translation. For instance, our devset contained the Spanish sentence "Estoy de acuerdo con el ponente en dos cuestiones ." The baseline translation given is "I agree with the rapporteur in to make .", and the SFR-scored translation given is "I agree with the rapporteur in both questions .". While both translations have the same GTM score against the gold standard "I agree with the rapporteur on two issues .", clearly, the second one is of far higher quality than the first. As we can see in Figure 4, the derivation over the substring "in both questions" gets a higher score than "in to make" when translated with the rescored grammar. In the baseline, "en dos cuestiones" is not translated as a whole unit – rather, the derivation of "el ponente en dos cuestiones" is decomposed into four subderivations, yielding "el" "ponente" "en" "dos cuestiones", where each of those is translated separately, into "∅" "the rapporteur" "in" and "to make". The SFR-scored grammar, however, outputs a different bilingual derivation. The source is decomposed into five sub-derivations, one for each word, and each word is translated separately. Then, the rescored target fragments set the proper target-side word order and select the target-side words that maximize the score of the subderivation covering the source span. We note that in this example, the score of translating "dos" to "make" was higher than the score of translating "dos" to "both". However, the higher level target fragment that composed the translation of "dos" together with the translation of "cuestiones" yielded a higher score when composing "both questions" rather than "to make".

7 Conclusions and Future Work

The results presented above indicate that augmenting the scoring mechanism with an accuracy-based measure is a promising direction for translation quality improvement. It gives us a statistically significant improvement over the baseline, and our analysis has indicated that the system is indeed making better decisions, moving us a step closer towards the goal of making translation decisions based on the hypothesis of the resulting transla-

tion's accuracy.

Now that we have demonstrated that translation quality can be improved by incorporating a measure of fragment quality into the scoring scheme, our immediate next step is to optimize our system so that we can scale up to significantly larger training and tuning sets, and determine whether the improvements we have noted carry over when the likelihood is computed from more data. Afterwards, we will implement a training scheme to maximize an accuracy-based objective function, for instance, by minimizing the difference between the scores of the highest-scoring derivation and the oracle derivations, in effect maximizing the score of the highest-scoring translation.

The rescoring method presented in this paper need not be limited to DOT. Fragments can be thought of as analogous to phrases in Phrase-Based SMT systems – we could implement a similar rescoring system for phrase-based systems, where we generate several candidate translations for source sentences in a tuning set, and score each phrase used against the phrases used in a set of oracles. More broadly, we could potentially take any statistical MT system, and compare the features of all candidates generated against those of oracle translations, and score those that are closer to the oracle higher than those further away.

Finally, by explicitly framing the translation problem as a search problem, where we are divorcing the inferences in the search space (i.e. the model) from the path we take to find the optimal inference according to some criterion (i.e. the scoring scheme), we can remove some of the variability when comparing two models or scoring mechanisms (Lopez, 2009).

Acknowledgements

This work is supported by Science Foundation Ireland (Grant No. 07/CE/I1142). We would like to thank the anonymous reviewers for their helpful comments and suggestions.

References

R. Bod. 2007. Unsupervised syntax-based machine translation: The contribution of discontiguous phrases. In *Proceedings of the 11th Machine Translation Summit*, pages 51–57, Copenhagen, Denmark.

P. F. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

F. J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176.

J. Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), Companion Volume*, pages 205–208, Sapporo.

M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, and I. Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia.

J. Goodman. 1996. Efficient algorithms for parsing the DOP model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 143–152, Philadelphia, PA.

M. Hearne and A. Way. 2003. Seeing the wood for the trees: Data-oriented translation. In *Proceedings of the Ninth Machine Translation Summit*, pages 165–172, New Orleans, LA.

M. Hearne and A. Way. 2006. Disambiguation strategies for data-oriented translation. In *Proceedings of the 11th Conference of the European Association for Machine Translation*, pages 59–68, Oslo, Norway.

M. Hearne. 2005. *Data-Oriented Models of Parsing and Translation*. Ph.D. thesis, Dublin City University, Dublin, Ireland.

M. Johnson. 2002. The DOP estimation method is biased and inconsistent. *Computational Linguistics*, 28(1):71–76, March.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, demonstation session*, pages 177–180, Prague, Czech Republic.

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of*

the Conference on Empirical Methods in Natural Language Processing, pages 388–395, Barcelona, Spain.

P. Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Machine Translation Summit X*, pages 79–86, Phuket, Thailand.

A. Lavie, K. Sagae, and S. Jayaraman. 2004. The significance of recall in automatic metrics for MT evaluation. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas*, pages 134–143, Washington, DC.

A. Lopez. 2009. Translation as weighted deduction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 532–540, Athens, Greece.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA.

A. Poutsma. 2000. Data-oriented translation. In *The 18th International Conference on Computational Linguistics*, pages 635–641, Saarbrücken, Germany.

C. Tillmann and T. Zhang. 2006. A discriminative global training algorithm for statistical MT. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 721–728, Sydney, Australia.

J. Tinsley, M. Hearne, and A. Way. 2009. Parallel treebanks in phrase-based statistical machine translation. In *Proceedings of the Tenth International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pages 318–331, Mexico City, Mexico.

J. Turian, L. Shen, and I. D. Melamed. 2003. Evaluation of machine translation and its evaluation. In *Proceedings of the Ninth Machine Translation Summit*, pages 386–393, New Orleans, LA.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530, Toulouse, France.

K. Yamada and K. Knight. 2002. A decoder for syntax-based statistical MT. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 303–310, Philadelphia, PA.

B. Zhao and S. Chen. 2009. A simplex armijo downhill algorithm for optimizing statistical machine translation decoding parameters. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 21–24, Boulder, Colorado.

Improved Statistical Machine Translation Using Monolingually-Derived Paraphrases

Yuval Marton,* **Chris Callison-Burch,†** and **Philip Resnik***

*Department of Linguistics and the CLIP Lab
at the Institute for Advanced Computer Studies (UMIACS)
University of Maryland College Park, MD 20742-7505, USA
{ymarton,resnik}@umiacs.umd.edu

†Computer Science Department, Johns Hopkins University
3400 N. Charles Street (CSEB 226-B) Baltimore, MD 21218
ccb@cs.jhu.edu

Abstract

Untranslated words still constitute a major problem for Statistical Machine Translation (SMT), and current SMT systems are limited by the quantity of parallel training texts. Augmenting the training data with paraphrases generated by pivoting through other languages alleviates this problem, especially for the so-called "low density" languages. But pivoting requires additional parallel texts. We address this problem by deriving paraphrases monolingually, using distributional semantic similarity measures, thus providing access to larger training resources, such as comparable and unrelated monolingual corpora. We present what is to our knowledge the first successful integration of a collocational approach to untranslated words with an end-to-end, state of the art SMT system demonstrating significant translation improvements in a low-resource setting.

1 Introduction

Phrase-based systems, flat and hierarchical alike (Koehn et al., 2003; Koehn, 2004b; Koehn et al., 2007; Chiang, 2005; Chiang, 2007), have achieved a much better translation coverage than word-based ones (Brown et al., 1993), but untranslated words remain a major problem in SMT. For example, according to Callison-Burch *et al.* (2006), a SMT system with a training corpus of 10,000 words learned only 10% of the vocabulary; the same system learned about 30% with a training corpus of 100,000 words; and even with a large training corpus of nearly 10,000,000 words it only reached about 90% coverage of the source vocabulary. Coverage of higher order n-gram levels is even harder. This problem plays a major part in reducing machine translation quality, as reflected by both automatic measures such as BLEU (Papineni et al., 2002) and human judgment tests. Improving translation coverage accurately is therefore important for SMT systems.

The first solution that might come to mind is to use larger parallel training corpora. However, current state-of-the-art SMT systems cannot learn from non-aligned corpora, while sentence-aligned parallel corpora (bitexts) are a limited resource (See Section 2 for discussion of automatically-compiled bitexts). Another direction might be to make use of non-parallel corpora for training. However, this requires developing techniques to extract alignments or translations from them, and in a sufficiently fast, memory-efficient, and scalable manner. One approach that can, in principle, better exploit both alignments from bitexts and make use of non-parallel corpora is the distributional collocational approach, e.g., as used by Fung and Yee (1998) and Rapp (1999). However, the systems described there are not easily scalable, and require pre-computation of a very large collocation counts matrix. Related attempts propose generating bitexts from comparable and "quasi-comparable" bilingual texts by iteratively bootstrapping documents, sentences, and words (Fung and Cheung, 2004), or by using a maximum entropy classifier (Munteanu and Marcu, 2005). Alignment accuracy remains a challenge for them.

Recent work has proposed augmenting the training data with paraphrases generated by pivoting through other languages (Callison-Burch et al., 2006; Madnani et al., 2007). This indeed alleviates the vocabulary coverage problem, especially for the so-called "low density" languages. However, these approaches still require bitexts where

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 381–390,
Singapore, 6-7 August 2009.

one side contains the original source language.

The paradigm described in this paper involves constructing monolingual distributional profiles (DPs; a.k.a. word association profiles, or co-occurrence vectors) of out-of-vocabulary words and phrases in the source language; then, generating paraphrase candidates from phrases that co-occur in similar contexts, and assigning them similarity scores. The highest ranking paraphrases are used to augment the translation phrase table. The table augmentation idea is similar to Callison-Burch *et al.*'s (Callison-Burch et al., 2006), but our proposed paradigm does not require using a limited resource such as parallel texts in order to generate paraphrases. Moreover, our proposed paradigm can, in principle, achieve large-scale acquisition of paraphrases with high semantic similarity. However, using parallel training texts in pivoting techniques offers the potential advantage of implicit translational knowledge, in the form of sentence alignments, while our approach is unguided in this respect. Therefore, we conducted experiments to find out how these relative advantages play out. We present here, to our knowledge for the first time, positive results of integrating distributional monolingually-derived paraphrases in an end-to-end state-of-the-art SMT system.

In the rest of this paper we discuss related work in Section 2, describe the distributional hypothesis and distributional profiles in Section 3, and present the monolingually-derived paraphrase generation system in Section 4. We report our experiments and results in Section 5, and conclude by discussing the implications and future research directions in Section 6.

2 Related Work

This is not the first to attempt to ameliorate the out-of-vocabulary (OOV) words problem in statistical machine translation, and other natural language processing tasks. This work is most closely related to that of Callison-Burch *et al.* (2006), who also translate source-side paraphrases of the OOV phrases. There, paraphrases are generated from bitexts of various language pairs, by "pivoting": translating the OOV phrases to an additional language (or languages) and back to the source language. The quality of these paraphrases is estimated by marginalizing translation probabilities to and from the additional language side(s) e, as follows: $p(f_2|f_1) = \sum_e p(e|f_1)p(f_2|e)$. A major disadvantage of their approach is that it relies on the availability of parallel corpora in other languages. While this works for English and many European languages, it is far less likely to help when translating from other source languages, for which bitexts are scarce or non-existent. Also, the pivoting approach is inherently noisy (in both the paraphrase candidates' correct sense, and their translational likelihood), and it is likely to fare poorly with out-of-domain translation. One advantage of the bitext-dependent pivoting approach is the use of the additional human knowledge that is encapsulated in the parallel sentence alignment. However, we argue that the ability to use much larger resources for paraphrasing should trump the human knowledge advantage.

More recently, Callison-Burch (2008) has improved performance of this pivoting technique by imposing syntactic constraints on the paraphrases. The limitation of such an approach is the reliance on a good parser (in addition to reliance on bitexts), but a good parser is not available in all languages, especially not in resource-poor languages. Another approach using a pivoting technique augments the human reference translation with paraphrases, creating additional translation "references" (Madnani et al., 2007). Both approaches have shown gains in BLEU score.

Barzilay and McKeown (2001) extract paraphrases from a monolingual parallel corpus, containing multiple translations of the same source. In addition to the parallel corpus usage limitations described above, this technique is further limited by the small size of such materials, which are even scarcer than the resources in the pivoting case.

Dolan *et al.* (2004) explore generating paraphrases by edit-distance and headlines of time- and topic-clustered news articles; they do not address the OOV problem directly, as their focus is sentence-level paraphrases; although they use a standard SMT measure, alignment error rate (AER), they only report results of the alignment quality, and not of an end-to-end SMT system. Much of the previous research largely focused on morphological analysis in order to reduce type sparseness; Callison-Burch *et al.* (2006) list some of the influential work in that direction.

Work that relies on the distributional hypothesis using bilingual comparable corpora (without the need for bitexts), typically uses a seed lexicon for "bridging" source language phrases

with their target languages paraphrases (Fung and Yee, 1998; Rapp, 1999; Diab and Finch, 2000). This approach is sometimes viewed as, or combined with, an information retrieval (IR) approach, and normalizes strength-of-association measures (see Section 3) with IR-related measures such as TF/IDF (Fung and Yee, 1998). To date, reported implementations suffer from scalability issues, as they pre-compute and hold in memory a huge collocation matrix; we know of no report of using this approach in an end-to-end SMT system.

Another approach aiming to reduce OOV rate concentrates on increasing parallel training set size without using more dedicated human translation (Resnik and Smith, 2003; Oard et al., 2003).

3 Collocational Profiles

The distributional hypothesis and distributional profiles. Natural language processing (NLP) applications that assume the distributional hypothesis (Harris, 1940; Firth, 1957) typically keep track of word co-occurrences in *distributional profiles* (a.k.a. *collocation vectors*, or *context vectors*). Each distributional profile DP_u (for some word u) keeps counts of co-occurrence of u with all words within a usually fixed distance from each of its occurrences (a *sliding window*) in some training corpus. More advanced profiles keep "strength of association" (SoA) information between u and each of the co-occurring words, which is calculated from the counts of u, the counts of the other word, their co-occurrence count, and the count of all words in the corpus (corpus size). The information on the other words with respect to u is typically kept in a vector whose dimensions correspond to all words in the training corpus. This is described in Equation (1), where V is the training corpus vocabulary:

$$DP_u = \{< w_i, SoA(u, w_i) > | u, w_i \in V\} \quad \text{for all } i \text{ s.t. } 1 \leq i \leq |V| \tag{1}$$

Semantic similarity between words u and v can be estimated by calculating the similarity (vector distance) between their profiles. Slightly more formally, the distributional hypothesis assumes that if we had access to the hypothetical true (psycholinguistic) semantic similarity function over word pairs, $semsim(u, v)$, then

$$\begin{aligned} &\forall u, v, w \in V, \\ &[semsim(u, v) > semsim(u, w)] \implies \\ &[psim(DP_u, DP_v) > psim(DP_u, DP_w)], \end{aligned} \tag{2}$$

where V is the language vocabulary, DP_{word} is the distributional profile of $word$, and $psim()$ is a 2-place vector similarity function (all further described below). Paraphrasing and other NLP applications that are based on the distributional hypothesis assume entailment in the reverse direction: the right-hand-side of Formula (2) (profile/vector similarity) entails the left-hand-side (semantic similarity).

The sliding window and word association (SoA) measures. Some researchers count *positional* collocations in a sliding window, i.e., the co-counts and SoA measures are calculated per relative position (e.g., for some word/token u, position 1 is the token immediately after u; position -2 is the token preceding the token that precedes u) (Rapp, 1999); other researchers use *non-positional* (which we dub here *flat*) collocations, meaning, they count all token occurrences within the sliding window, regardless of their positions in it relative to u (McDonald, 2000; Mohammad and Hirst, 2006). We use here flat collocations in a 6-token sliding window. Beside simple co-occurrence counts within sliding windows, other SoA measures include functions based on TF/IDF (Fung and Yee, 1998), mutual information (PMI) (Lin, 1998), conditional probabilities (Schuetze and Pedersen, 1997), chi-square test, and the log-likelihood ratio (Dunning, 1993).

Profile similarity measures. A profile similarity function $psim(DP_u, DP_v)$ is typically defined as a two-place function, taking vectors as arguments, each vector representing a distributional profile of some word u and v, respectively, and whose cells contain the SoA of u (or v) with each word ("collocate") in the known vocabulary. Similarity can be (and have been) estimated in several ways, e.g., the cosine coefficient, the Jaccard coefficient, the Dice coefficient, and the City-Block measure. The formula for the cosine function for similarity measure is given in Eq. (3):

$$psim(DP_u, DP_v) = cos(DP_u, DP_v) = \frac{\sum_{w_i \in V} SoA(u, w_i) SoA(v, w_i)}{\sqrt{\sum_{w_i \in V} SoA(u, w_i)^2} \sqrt{\sum_{w_i \in V} SoA(v, w_i)^2}} \quad (3)$$

In principle, any SoA can be used with any profile similarity measure. However, in practice, only some SoA/similarity measure combinations do well, and finding the best combination is still more art than science. Some successful combinations are cos_{CP} (Schuetze and Pedersen, 1997), Lin_{PMI} (Lin, 1998), $City_{LL}$ (Rapp, 1999), and Jensen–Shannon divergence of conditional probabilities (JSD_{CP}). We use here cosine of log-likelihood vectors (McDonald, 2000).

Phrasal distributional profiles. Word DPs can be generalized to phrasal DPs, simply by counting words that co-occur within a sliding window around the target phrase's occurrences (i.e., counting occurrences of words up to 6 words before or after the target phrase). For example, when building a DP for the target phrase *counting words* in the previous sentence, then *simply* is in relative position -2, and *sliding* is in relative position 5. Searching for similar phrasal DPs poses an additional challenge over the word DP case (see Section 4), but there is no additional difficulty in building the phrasal profile itself as described above. In preliminary experiments we found no gain in using phrasal collocates (i.e., count how many times a *phrase* of more than one word co-occurs in a sliding window around the target word/phrase).

4 Searching and Scoring Phrasal Paraphrases

The system design is as follows: upon receiving OOV phrase phr, build distributional profile DP_{phr}. Next, gather contexts: for each occurrence of phr, keep surrounding (left and right) context $L\_\_R$. For each such context, gather paraphrase candidates X which occur between L and R in other locations in the training corpus, i.e., all X such that LXR occur in the corpus. Finally, rank all candidates X, by building distributional profile DP_X and measuring profile similarity between DP_X and DP_{phr}, for each X. Output k-best candidates above a certain similarity score threshold. The rest of this section describes this system in more detail.

Build phrasal profile DP_{phr}. Build a profile of all word collocates, as described in Section 3. Use sliding window of size $MaxPos = 6$. If phr is very frequent (above some threshold of t occurrences), uniformly sample only t occurrences, multiplying the gathered co-counts by factor of $count(phr)/t$. We set $t = 10000$.

Gather context. The challenge in choosing the relevant context is this: if it is very short and/or very frequent (e.g., "the __ is"), then it might not be very informative, in the sense that many words can appear in that context (in this example, practically any noun); however, if it is too long (too specific), then it might not occur enough times elsewhere (or not at all) in the training corpus. Therefore, to balance between these two extremes, we use the following heuristics. Start small: Start with setting the left part of the context L to be a single word/token to the left of phrase phr. If it is stoplisted, append the next word to the left (now having a bigram left context instead of a unigram), and repeat until the left context is not in the stoplist. Repeat similarly for R, the context to the right of phr. Add the resulting $L\_\_R$ context to a context list. We stoplist "promiscuous" words, i.e., those that have more than $StoplistThreshold$ collocates in the training corpus, using the above $MaxPos$ parameter value. We also stoplist bigrams which occur more than t times and comprise solely from stoplisted unigrams.

Gather candidates. For each gathered context in the context list, gather all paraphrase candidate phrases X that connect left hand side context L with right hand side context R, i.e., gather all X such that the sequence LXR occurs in the corpus. In practice, to keep search complexity low, limit X to be up to length $MaxPhraseLen$. Also, to further speed up runtime, we uniformly sample the context occurrences.

Rank candidates. For each candidate X, build distributional profile DP_X, and evaluate $psim(DP_{phr}, DP_X)$.

Output k-best candidates. Output k-best paraphrase candidates for phrase phr, in descending order of similarity. We set $k = 20$. Filter out paraphrases with score less than $minScore$.

5 Experiment

We examined the application of the system's paraphrases to handling unknown phrases when translating from English into Chinese (E2C) and from Spanish into English (S2E). For all baselines we used the phrase-based statistical machine translation system Moses (Koehn et al., 2007), with the default model features, weighted in a log-linear framework (Och and Ney, 2002). Feature weights were set with minimum error rate training (Och, 2003) on a development set using BLEU (Papineni et al., 2002) as the objective function. Test results were evaluated using BLEU and TER (Snover et al., 2005). The phrase translation probabilities were determined using maximum likelihood estimation over phrases induced from word-level alignments produced by performing Giza++ training (Och and Ney, 2000) on both source and target sides of the parallel training sets. When the baseline system encountered unknown words in the test set, its behavior was simply to reproduce the foreign word in the translated output.

The paraphrase-augmented systems were identical to the corresponding baseline system, with the exception of additional (paraphrase-based) translation rules, and additional feature(s). Similarly to Callison-Burch *et al.* (2006), we added the following feature:

$$h(e,f) = \begin{cases} psim(DP_{f'}, DP_f) & \text{If phrase table entry } (e,f) \\ & \text{is generated from } (e,f') \\ & \text{using monolingually-} \\ & \text{derived paraphrases.} \\ 1 & \text{Otherwise,} \end{cases} \quad (4)$$

Note that it is possible to construct a new translation rule from f to e via more than one pair of source-side phrase and its paraphrase; e.g., if f_1 is a paraphrase of f, and so is f_2, and both f_1, f_2 translate to the same e, then both lead to the construction of the new rule translating f to e, but with potentially different feature scores.

In order to eliminate this duplicity and leverage over these alternate paths which can be used to increase our confidence level in the new rule, we did the following: For each paraphrase f of some source-side phrases f_i, with respective similarity scores $sim(f_i, f)$, we calculated an aggregate score $asim$ with a "quasi-online-updating" method as follows: $asim_i = (1 - asim_{i-1})sim(f_i, f)$, where $asim_0 = 0$. The aggregate score $asim$ is updated in an "online" fashion with each pair f_i, f as they are processed, but only the final $asim_k$ score is used, after all k pairs have been processed. Simple arithmetics can show that this method is insensitive to the order in which the paraphrases are processed. We only augment the phrase table with a single rule from f to e, and in it are the feature values of the phrase f_i for which the score $sim(f_i, f)$ was the highest.

5.1 English-to-Chinese Translation

For the English-Chinese (E2C) baseline system, we trained on the LCD Sinorama and FBIS tests (LCD2005T10 and LCD2003E14), and segmented the Chinese side with the Stanford Segmenter (Tseng et al., 2005). After tokenization and filtering, this bitext contained 231,586 lines (6.4M + 5.1M tokens). We trained a trigram language model on the Chinese side. We then split the bitext to 32 even slices, and constructed a reduced set of about 29,000 lines (sentences) by using only every eighth slice. The purpose of creating this subset model was to simulate a resource-poor language. See Table 1.

Set	# Tokens Source+Target
E2C 29K	0.8 + 0.6
E2C Full	6.4 + 5.1
bnc+apw	187
S2E 10K	0.3 + 0.3
S2E 20K	0.6 + 0.6
S2E 80K	2.3 + 2.3
wmt09	84
wmt09+acquis	139
wmt09+acquis+afp	402

Table 1: Training set sizes (million tokens).

For development, we used the Chinese-English NIST MT 2005 evaluation set, taking one of the English references as source, and the Chinese source as a single reference translation. We tested the system using the English-Chinese NIST MT evaluation 2008 test set with its four reference translations.

We augmented the E2C baseline models with paraphrases generated as described above, training on the British National Corpus (BNC) v3 (Burnard, 2000) and the first 3 million lines of the English Gigaword v2 APW, totaling 187M terms after tokenization, and number and punctuation removal. We generated paraphrases for phrases up to six tokens in length, and used an ar-

bitrary similarity threshold of $minScore = 0.3$. We experimented with three variants: adding a single additional feature for all paraphrases (*1-6grams*); using only paraphrases of unigrams (*1grams*); and adding two features, one only sensitive to unigrams, and the other only to the rest (*1 + 2-6grams*). All features had the same design as described in Section 5, each had an associated weight (as all other features), and all feature weights in each system, including the baseline, were tuned using a separate minimum error rate training for each system.

Results are shown in Table 2. For the E2C systems, for which we had four reference translations for the test set, we used shortest reference length, and used the NIST-provided script to split the output words to Chinese characters before evaluation. Statistical significance for the BLEU results were calculated using Koehn's (Koehn, 2004) pair-wise bootstrapping test with 95% confidence interval.

On the E2C 29,000-line subset, the augmented system had a significant 1.7 BLEU points gain over its baseline. On the full size model, results were negative. Note that our E2C full size baseline is reasonably strong: Its character-based BLEU score is slightly higher than the JHU-UMD system that participated in the NIST 2008 MT evaluation (constrained training track), although we used a subset of that system's training materials, and a smaller language model. Results there ranged from 15.69 to 30.38 BLEU (ignoring a seeming outlier of 3.93).

5.2 Spanish-to-English Translation

In order to to permit a more direct comparison with the pivoting technique, we also experimented with Spanish to English (S2E) translation, following Callison-Burch *et al.* (2006). For baseline we used the Spanish and English sides of the Europarl multilingual parallel corpus (Koehn, 2005), with the standard training, development, and test sets. We created training subset models of 10,000, 20,000, and 80,000 aligned sentences, as described in Callison-Burch *et al.* (2006). For better comparison with their pivoting system, we used the same 5-gram language model, development and test sets: For development, we used the Europarl dev2006 Spanish and English sides, and for testing we used the Europarl 2006 test set.

We trained the Spanish paraphrase generation system on the Spanish corpora available from

dataset	E2C model	BLEU	TER
29k	baseline	15.21	90.354
29k	1grams	16.87***	90.370
29k	1-6grams	16.54***	90.376
29k	1 + 2-6grams	**16.88*** **	**90.349**
Full	baseline	**22.17**	**90.398**
Full	1grams	21.64***	90.459
Full	1-6grams	21.75	90.421
Full	1 + 2-6grams	21.39***	90.433

Table 2: E2C Results: character-based BLEU and TER scores. All models have one additional feature over baseline, except for the "1 + 2-6" models that have one feature for unigrams and another feature for bigrams to 6-grams. Paraphrases with score $<$ .3 were filtered out. *** = significance test over baseline with $p < 0.0001$, using Koehn's (2004) pair-wise bootstrap resampling test for BLEU with 95% confidence interval.

Paraphrase	Score
Source: *deal*	
agreement	0.56
accord	0.53
talks	0.45
contract	0.42
peace deal	0.33
merger	0.32
agreement is	0.30
Source: *fall*	
rise	0.87
slip	0.82
tumbled today	0.68
fell today	0.67
tumble	0.65
fall tokyo ap stock prices fell	0.56
are mixed	0.54
Source: *to provide any other*	
to give any	0.74
to give further	0.70
to provide any	0.68
to give any other	0.62
to provide further	0.61
to provide other	0.53
to reveal any	0.52
to provide any further	0.48
to disclose any	0.47
to publicly discuss the	0.43
Source: *we have a situation that*	
uncontroversial question about our	0.66
obviously with the developments this morning	0.65
community staffing of community centres	0.64
perhaps we are getting rather impatient	0.63
er around the inner edge	0.60
interested in going to the topics	0.60
and that is the day that	0.60
as a as a final point	0.59
left which it may still have	0.56

Table 3: English paraphrases from E2C 29K-bitext systems.

the EACL 2009 Fourth Workshop on Statistical Machine Translation:[1] the Spanish side of the Europarl-v4, news training 2008, and news commentary 2009. We also re-trained adding the JRC-Acquis-v3 corpus[2] to the paraphrase training set, and then adding also the LDC Spanish Gigaword (LDC2006T12) and truncating the resulting corpus after the first 150M lines. We lowercased these training sets, tokenized and removed punctuation marks and numbers, and this resulted in training set sizes as detailed in Table 1. We generated paraphrases for phrases up to four tokens in length, and used two arbitrary similarity thresholds of $minScore = 0.3$ (as in the E2C experiments), and 0.6, for enforcing only higher precision paraphrasing.

We experimented with these variants: a single feature for all paraphrase (*1-4grams*); using only paraphrases of unigrams (*1grams*); and using two features: one only sensitive to unigrams and bigrams, and the other to the rest (*1-2 + 3-4grams*).

Results are shown in Table 4. We used BLEU over lowercased outputs to evaluate all S2E systems, and Koehn's significance test as above.

On the S2E 10,000-line subset, both the *1grams* and *1-4grams* models achieved significant gains of .4 BLEU points over the baseline. We concluded from a manual evaluation of the 10,000-line models that the two major weaknesses of the baseline system were (not surprisingly) number of untranslated (OOV) words / phrases, followed by number of superfluous words / phrases.

On the larger subset models, no system significantly outperformed the baseline. Note that our S2E baselines' scores are higher than those of Callison-Burch *et al.* (2006), since we evaluate lowercased outputs, instead of recased ones.

6 Discussion and Future Work

We have shown that monolingually-derived paraphrases, based on distributional semantic similarity measures over a source-language corpus, can improve the performance of statistical machine translation (SMT) systems. Our proposed method has the advantage of not relying on bitexts in order to generate the paraphrases, and therefore gives access to large amounts of monolingual training data, for which creating bitexts of equivalent size is generally unfeasible. We haven't trained our system on nearly as large a corpus as it can handle, and indeed we see this as a natural next step.

Results support the assumption that a larger monolingual paraphrase training set yields better paraphrases: our S2E *1-4grams* model performed significantly better than baseline when using *wmt09+aquis* for paraphrasing, but when only using *wmt09*, the model had a smaller advantage that did not reach significance. However, for the S2E *1grams* model, there was a slight decrease in performance when switching paraphrasing corpus from *wmt09+aquis* to *wmt09+aquis+afp*. This effect might be due to the genre or unbalanced content of the additional corpus, or perhaps it is the case that in this corpus size, paraphrases of higher-level ngrams benefitted from the additional text much more than paraphrases of unigrams did. The two rightmost columns in Table 5 show that although Spanish monolingual paraphrases for the unigram *baile* improve when using the larger corpus, (e.g., *danza* and *un balie* become the third and fourth top candidates, pushing much worse candidates far down the list), the two top paraphrase candidates remained unchanged. However, for the 4gram *a favor del informe*, antonymous candidates, which are bad and misleading for translation, are pushed down from the top first and third spots by synonymous, better candidates. Table 3 contains additional examples of good and bad top paraphrase candidates, also in English. Paraphrases of phrases seem to be of lower quality than those of unigrams, as can be seen at the bottom of the table.

These results also show that our method is especially useful in settings involving low-density languages or special domains: The smaller subset models, emulating a resource-poor language situation, show higher gains than larger models (which are supersets of the smaller subset models), when augmented with paraphrases derived from the same paraphrase training set. This was validated in two very different language pairs: English to Chinese, and Spanish to English. We believe that larger monolingual training sets for paraphrasing can help languages with richer resources, and we intend to explore this too.

Although the gains in the Spanish-English subsets are somewhat smaller than the pivoting technique reported in Callison-Burch *et al.* (2006), e.g., .7 BLEU for the 10k subset, we take these results as a proof of concept that can yield better

[1] http://www.statmt.org/wmt09

[2] http://wt.jrc.it/lt/Acquis

bitext	mono.corp.	features	minScore	BLEU	TER
10k	(baseline)	–	–	23.78	62.382
10k	wmt09	1-4grams	.6	23.81	
10k	wmt09	1-2+3-4gr	.6	23.92	62.202
10k	wmt09+aquis	1-4grams	.6	**24.13*****	**61.739**
10k	wmt09+aquis	1grams	.6	24.11	61.979
20k	(baseline)	–	–	24.68	62.333
20k	wmt09+aquis	1-4grams	.6	**24.75**	**61.528**
80k	(baseline)	–	–	**27.89**	57.977
80k	wmt09+aquis	1-4grams	.6	27.82	**57.906**
10k	wmt09+aquis	1grams	.3	**24.11**	61.979
10k	wmt09+aquis+afp	1grams	.3	23.97	61.974
20k	wmt09+aquis+afp	1grams	.3	**24.77**	61.276
80k	wmt09+aquis+afp	1grams	.3	27.84***	**57.781**

Table 4: S2E Results: Lowercase BLEU and TER. Paraphrases with score < *minScore* were filtered out. *** = significance test over baseline with $p < 0.0001$, using Koehn's (2004) pair-wise bootstrap test for BLEU with 95% confidence interval.

pivot	wmt09+acquis	wmt09+acquis+afp
Source: *baile*		
danza	el baile	el baile
bailar	baile y	baile y
a	de david palomar y la	danza
dans	viejo como quien se acomoda una	un baile
empresa	por julián estrada el tercero de	teatro
coro	al baile a la	baloncesto el cine
Source: *a favor del informe*		
a favor de este informe	en contra del informe	favor del informe
favor del informe	a favor de este informe	en contra del informe
el informe	en contra de este informe	a favor de este informe
a favor	a favor de la resolución	en contra de este informe
por el informe	a favor de esta resolución	en contra de la resolución
al informe	a favor del informe del señor	a favor del informe del sr.
su	a favor del informe del sr.	en contra del informe del sr.
del informe	en contra de la propuesta	a favor del excelente informe
de este informe	contra el informe	a favor del informe deprez

Table 5: Comparison of Spanish paraphrases: by pivoting, and by two monolingual corpora. Ordered from best to worst score.

system	example
source	cuando escucho las distintas intervenciones , creo que quienes afirman que deberíamos analizar nuestras prioridades y limitar el número de objetivos que queremos conseguir , están en lo cierto .
reference	when i listen to the various comments made , i find myself agreeing with those who recommend that we take a look at our priorities and then limit the number of aims we want to achieve
baseline	**escucho** when the various speeches, i believe that those who **afirman** that we should our environmental **limitar** priorities and the number of objectives we want to achieve, are in this way.
pivoting (MW)	when i can **hear** the various speeches , i believe that those people that we should look at our priorities and to **limit** the number of objectives we want to achieve , are in fact .
wmt09+acquis .1-4grams	**escucho** when the various speeches, i believe that those who **claiming** that we should environmental **limitar** our priorities and the number of objectives we want to achieve, are on the way.
wmt09+acquis .1grams	**escucho** when the various speeches, i believe that those who **considered** that we should our environmental priorities and **reducing** the number of objectives we want to achieve, are on the way.
wmt09+acquis+afp .1grams	**escucho** when the various speeches, i believe that those who **say** that we should our environmental priorities and **reduce** the number of objectives we want to achieve, are on the way.

Table 6: S2E translation examples on 10k-bitext systems. Some translation differences are in bold.

gains with larger monolingual training sets. Pivoting techniques (translating back and forth) rely on limited resources (bitexts), and are subject to shifts in meaning due to their inherent double translation step. In contrast, large monolingual resources are relatively easy to collect, and our system involves only a single translation/paraphrasing step per target phrase. Table 5 also shows an exemplar comparison with the pivoting paraphrases used in Callison-Burch *et al.* (2006). It seems that the pivoting paraphrases might suffer more from having frequent function words as top candidates, which might be a by-product of their alignment "promiscuity". However, the top antonymous candidate problem seems to mainly plague the monolingual distributional paraphrases (but improves with larger corpora). See also Table 6.

The paraphrase quality remains an issue with this method (as with all other paraphrasing methods). Some possible ways of improving it, besides using larger corpora, are: using syntactic information (Callison-Burch, 2008), using semantic knowledge such as thesaurus or WordNet to perform word sense disambiguation (WSD) (Resnik, 1999; Mohammad and Hirst, 2006), improving the similarity measure, and refining the similarity threshold. We would like to explore ways of incorporating syntactic knowledge that do not sacrifice coverage as much as in Callison-Burch (2008); incorporating semantic knowledge to disambiguate phrasal senses; using context to help sense disambiguation (Erk and Padó, 2008); and optimizing the similarity threshold for use in SMT, for example on a held-out dataset: too high a threshold reduces coverage, while too low a threshold results in bad paraphrases and translation.

The method presented here is quite general, and therefore different similarity measures, including other corpus-based ones, can be plugged in to generate paraphrases. We are looking into using DPs with word-sense disambiguation: Since it has been shown that similarity is often judged by the semantic distance of the closest senses of the two target words (Mohammad and Hirst, 2006), and that paraphrases generated this way are likely to be of higher quality (Marton et al., 2009), hence it is also likely that the overall performance of an SMT system using them will also improve further.

One potential advantage of using bitexts for paraphrase generation is the usage of implicit human knowledge, i.e., sentence alignments. The concern that not using this knowledge would turn out detrimental to the performance of SMT systems augmented by paraphrases as described here was largely put to rest, as our method improved the tested subset SMT systems' quality.

Acknowledgments

Many thanks to Chris Dyer for his help with the E2C set, and to Adam Lopez for his implementation of pattern matching with Suffix Array. This research was partially supported by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-2-001 and NSF award 0838801, by the EuroMatrixPlus project funded by the European Commission, and by the US National Science Foundation under grant IIS-0713448. The views and findings are the authors' alone.

References

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

P.F. Brown, S.A.D. Pietra, V.J.D. Pietra, and R.L. Mercer. 1993. The mathematics of statistical machine translation. *Computational Linguistics*, 19(2):263–313.

Lou Burnard. 2000. *Reference Guide for the British National Corpus*. Oxford University Computing Services, Oxford, England, world edition edition.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings NAACL-2006*.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of EMNLP 2008*, Waikiki, Hawai'i.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL-05*, pages 263–270.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Mona Diab and Steve Finch. 2000. A statistical word-level translation model for comparable corpora. In *Proceedings of the Conference on Content-Based Multimedia Information Access (RIAO)*.

B. Dolan, C. Quirk, and C. Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics of the Association for Computational Linguistics*, Geneva, Switzerland.

T. Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2086)*, pages 897–906, Honolulu, HI.

John R. Firth. 1957. A synopsis of linguistic theory 1930Ű55. *Studies in Linguistic Analysis*, (special volume of the Philological Society):1–32. Distributional Hypothesis.

Pascale Fung and Percy Cheung. 2004. Multi-level bootstrapping for extracting parallel sentences from a quasi-comparable corpus. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1051, Geneva, Switzerland. Association for Computational Linguistics.

Pascale Fung and Lo Yuen Yee. 1998. An ir approach for translating new words from nonparallel, comparable texts. In *Proceedings of COLING-ACL98*, pages 414–420, Montreal, Canada.

Zellig S. Harris. 1940. Review of louis h. gray, foundations of language (new york: Macmillan, 1939). *Language*, 16(3):216Ű231.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 127–133.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL), demonstration session*, Prague, Czech Republic.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP*.

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

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

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304, San Francisco, CA.

Nitin Madnani, Necip Fazil Ayan, Philip Resnik, and Bonnie Dorr. 2007. Using paraphrases for parameter tuning in statistical machine translation. In *Proceedings of the ACL Workshop on Statistical Machine Translation.*

Yuval Marton, Saif Mohammad, and Philip Resnik. 2009. Estimating semantic distance using soft semantic constraints in knowledge-source / corpus hybrid models. In *Procedings of EMNLP*, Singapore.

S. McDonald. 2000. *Environmental determinants of lexical processing effort*. Ph.D. thesis, University of Edinburgh.

Saif Mohammad and Graeme Hirst. 2006. Distributional measures of concept-distance: A task-oriented evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2006)*, Sydney, Australia.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504.

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

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the ACL*, pages 440–447.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, John Henderson, and Florence Reeder. 2002. Corpus-based comprehensive and diagnostic MT evaluation: Initial Arabic, Chinese, French, and Spanish results. In *Proceedings of the ACL Human Language Technology Conference*, pages 124–127, San Diego, CA.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th Annual Conference of the Association for Computational Linguistics.*, pages 519–525.

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

Philip Resnik. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research (JAIR)*, 11:95–130.

Hinrich Schuetze and Jan O. Pedersen. 1997. A cooccurrence-based thesaurus and two applications to information retreival. *Information Processing and Management*, 33(3):307Ű318.

Matthew Snover, Bonnie J. Dorr, Richard Schwartz, John Makhoul, Linnea Micciulla, and Ralph Weischedel. 2005. A study of translation error rate with targeted human annotation. Technical Report LAMP-TR-126, CS-TR-4755, UMIACS-TR-2005-58, University of Maryland, July, 2005.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter. In *Fourth SIGHAN Workshop on Chinese Language Processing*.

A Comparison of Model Free versus Model Intensive Approaches to Sentence Compression

Tadashi Nomoto
National Institute of Japanese Literature
10-3 Midori Tachikawa
Tokyo 190-0014 Japan
nomoto@acm.org

Abstract

This work introduces a model free approach to sentence compression, which grew out of ideas from Nomoto (2008), and examines how it compares to a state-of-art model intensive approach known as Tree-to-Tree Transducer, or T3 (Cohn and Lapata, 2008). It is found that a model free approach significantly outperforms T3 on the particular data we created from the Internet. We also discuss what might have caused T3's poor performance.

1 Introduction

While there are a few notable exceptions (Hori and Furui, 2004; Yamagata et al., 2006), it would be safe to say that much of prior research on sentence compression has been focusing on what we might call 'model-intensive approaches,' where the goal is to mimic human created compressions as faithfully as possible, using probabilistic and/or machine learning techniques (Knight and Marcu, 2002; Riezler et al., 2003; Turner and Charniak, 2005; McDonald, 2006; Clarke and Lapata, 2006; Cohn and Lapata, 2007; Cohn and Lapata, 2008; Cohn and Lapata, 2009). Because of this, the question has never been raised as to whether a model free approach – where the goal is not to model what humans would produce as compression, but to provide compressions just as useful as those created by human – will offer a viable alternative to model intensive approaches. This is the question we take on in this paper.[1]

[1]One caveat would be in order. By *model free approach*, we mean a particular approach which does not furnish any parameters or weights that one can train on human created compressions. An approach is said to be *model-intensive* if it does. So as far as the present paper is concerned, we might do equally well with a mention of 'model free' ('model-intensive') replaced with 'unsupervised' ('supervised'), or 'non-trainable' ('trainable').

An immediate benefit of the model-free approach is that we could free ourselves from the drudgery of collecting gold standard data from humans, which is costly and time-consuming. Another benefit is intellectual; it opens up an alternative avenue to addressing the problem of sentence compression hitherto under-explored.

Also breaking from the tradition of previous research on sentence compression, we explore the use of naturally occurring data from the Internet as the gold standard. The present work builds on and takes further an approach called 'Generic Sentence Trimmer' (GST) (Nomoto, 2008), demonstrating along the way that it could be adapted for English with relative ease. (GST was originally intended for Japanese.) In addition, to get a perspective on where we stand with this approach, we will look at how it fares against a state-of-the-art model intensive approach known as 'Tree-to-Tree Transducer' (T3) (Cohn, 2008), on the corpus we created.

2 Approach

Nomoto (2008) discusses a two-level model for sentence compression in Japanese termed 'Generic Sentence Trimmer' (GST), which consists of a component dedicated to producing grammatical sentences, and another to reranking sentences in a way consistent with gold standard compressions. For the convenience's sake, we refer to the generation component as 'GST/g' and the ranking part as 'GST/r.' The approach is motivated largely by the desire to make compressed sentences linguistically fluent, and what it does is to retain much of the syntax of the source sentence as it is, in compression, which stands in contrast to Filippova and Strube (2007) and Filippova and Strube (2008), who while working with dependency structure (as we do), took the issue to be something that can be addressed by selecting and reordering constituents that are deemed relevant.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 391–399,
Singapore, 6-7 August 2009.

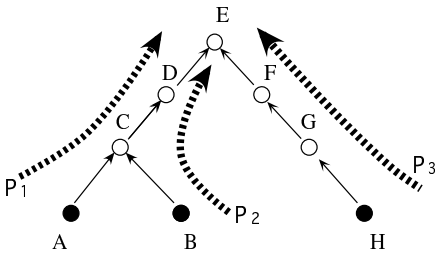


Figure 1: Dependency Structure for 'ABCDEFGH'

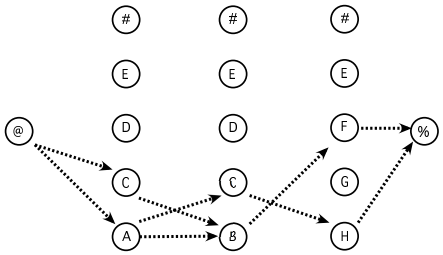


Figure 2: TDP Trellis and POTs.

Getting back to GST, let us consider a sentence,

(1) The bailout plan was likely to depend on private investors to purchase the toxic assets that wiped out the capital of many banks.

Among possible compressions GST/g produces for the sentence are:

(2)

a. The bailout plan was likely to depend on private investors to purchase the toxic assets.

b. The bailout plan was likely to depend on private investors.

c. The bailout plan was likely to depend on investors.

d. The bailout plan was likely.

Notice that they do not differ much from the source sentence (1), except that they get some of the parts chopped off. In the following, we talk about how this could done systematically.

3 Compression with Terminating Dependency Paths

One crucial feature of GST is the notion of *Terminating Dependency Paths* or TDPs, which enables us to factorize a dependency structure into a set of independent fragments. Consider string s = ABCDEFGH with a dependency structure as shown in Figure 1. We begin by locating terminal nodes, i.e., those which have no incoming edges, depicted as filled circles in Figure 1. Next we find a dependency (singly linked) path from each terminal node to the root (labeled E). This would give us three paths p_1 = A-C-D-E, p_2 = B-C-D-E, and p_3 = H-G-F-E (represented by dashed arrows in Figure 1).

Given TDPs, we set out to find a set $\mathcal{T}$ of all suffixes for each TDP, including an empty string, which would look like:

$$\mathcal{T}(p_1) = \{\langle \text{A C D E}\rangle, \langle \text{C D E}\rangle, \langle \text{D E}\rangle, \langle \text{E}\rangle, \langle\rangle\}$$
$$\mathcal{T}(p_2) = \{\langle \text{B C D E}\rangle, \langle \text{C D E}\rangle, \langle \text{D E}\rangle, \langle \text{E}\rangle, \langle\rangle\}$$
$$\mathcal{T}(p_3) = \{\langle \text{G F E}\rangle, \langle \text{F E}\rangle, \langle \text{E}\rangle, \langle\rangle\}$$

Next we combine suffixes, one from each set $\mathcal{T}$, while removing duplicates if any. Combining, for instance, $\langle \text{A C D E}\rangle \in \mathcal{T}(p_1)$, $\langle \text{C D E}\rangle \in \mathcal{T}(p_2)$, and $\langle \text{G F E}\rangle \in \mathcal{T}(p_3)$, would produce {A C D E G F}, which we take to correspond to a string ACDEGF, a short version of s.

As a way of doing this systematically, we put TDPs in a trellis format as in Figure 2, each file representing a TDP, and look for a path across the trellis, which we call 'POT.' It is easy to see that traveling across the trellis (while keeping record of nodes visited), gives you a particular way in which to combine TDPs: thus in Figure 2, we have three POTs, C-B-F, A-C-H, and A-B-F, giving rise to BCDEF, ACDEFGH, and ABCDEF, respectively (where '&' denotes a starting node, '%' an ending node, and '#' an empty string). Note that the POT in effect determines what compression we get.

Take for instance a POT C-B-F. To get to a compression, we first expand C-B-F to get $\{\langle \underline{\text{C}}\ \text{D E}\rangle_1, \langle \underline{\text{B}}\ \text{C D E}\rangle_2, \langle \underline{\text{F}}\ \text{E}\rangle_3\}$ (call it $\mathcal{E}$(C-B-F)). (Note that each TDP is trimmed to start with a node at a corresponding position of the POT.) Next we take a union of TDPs treating them as if they were sets: thus $\bigcup \mathcal{E}(\text{C-B-F}) = \{\text{B C D E F}\}$ = BCDEF.

4 N-Best Search over TDP Trellis

An obvious problem of this approach, however, is that it spawns hundreds of thousands of possible POTs. We would have as many as $5^3 = 125$ of them for the eight-character long string in Figure 1.

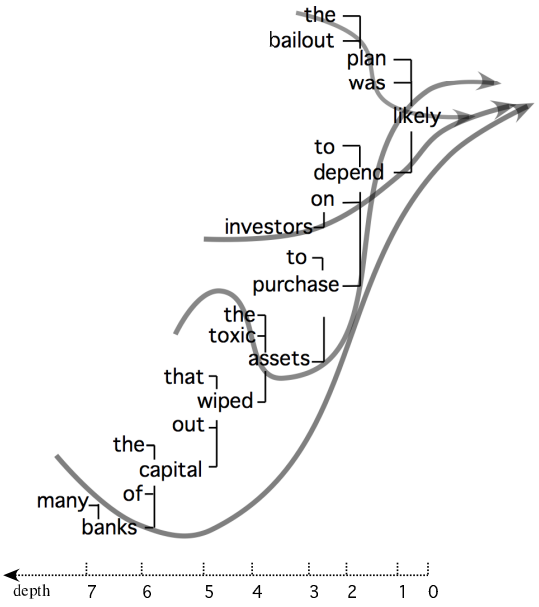


Figure 3: Dependency Structure

What we propose to deal with this problem is to call on a particular ranking scheme to discriminate among candidates we get. Our scheme takes the form of Equation 3 and 4.

$$W(x) = \mathrm{idf}(x) + \exp(-depth(x)) \quad (3)$$

$$S(\mathbf{p}) = \sum_{x_0,\ldots x_n \in \mathcal{E}(\mathbf{p})} W(x_i) \quad (4)$$

$depth(x)$ indicates the distance between x and the root, measured by the number of edges one need to walk across to reach the root from x. Figure 3 shows how the depth is gauged for nodes in a dependency structure. $\mathrm{idf}(x)$ represents the log of the inverse document frequency of x. The equations state that the score S of a POT $\mathbf{p}$ is given as the sum of weights of nodes that comprise $\bigcup \mathcal{E}(\mathbf{p})$.

Despite their being simple, equations 3 and 4 nicely capture our intuition about the way the trimming or compression should work, i.e., that the deeper we go down the tree, or the further away you are from the main clause, the less important information becomes. Putting aside $\mathrm{idf}(x)$ for the moment, we find in Figure 3, $W(assets) > W(capital) > W(banks) > W(many)$. Also depicted in the figure are four TDPs starting with *many*, *the* (preceding *toxic*), *investors*, and *the* (preceding *bailout*).

Finally, we perform a best-first search over the trellis to pick N highest scoring POTs, using Forward DP/Backward A* (Nagata, 1994), with the evaluation function given by Equation 4. We found that the beam search, especially when used with a small width value, does not work as well as the best first search as it tends to produce very short sentences due to its tendency to focus on inner nodes, which generally carry more weights compared to those on the edge. In the experiments described later, we limited the number of candidates to explore at one time to 3,000, to make the search computationally feasible.

Table 1: Drop-me-not rules. A '|' stands for *or*. 'a:b' refers to an element which has both a and b as attributes. Relation names such as *nsubj*, *aux*, *neg*, etc., are from de Marneffe et al. (2006).

R1.	VB	$\Rightarrow$	*nsubj* | *aux* | *neg* | *mark*
R2.	VB	$\Rightarrow$	WDT | WRB
R3.	JJ	$\Rightarrow$	*cop*
R4.	NN	$\Rightarrow$	*det* | *cop*
R5.	NN	$\Rightarrow$	*poss:*WP (='whose')
R6.	$\star$	$\Rightarrow$	*conj* & *cc*

5 'Drop-me-not' Rules

Simply picking a path over the TDP trellis (POT), however, does not warrant the grammaticality of the tree that it generates. Take for instance, a dependency rule, '*likely*←*plan*, *was*, *depend*,' which forms part of the dependency structure for sentence (1). It gives rise to three TDPs, ⟨*plan*, *likely*⟩, ⟨*was*, *likely*⟩, and ⟨*depend*, *likely*⟩. Since we may arbitrarily choose either of the two tokens in each TDP with a complete disregard for a syntagmatic context that each token requires, we may end up with sequences such as '*plan likely*,' '*plan was likely*,' or '*plan likely depend*' (instances of a same token are collapsed into one). This would obviously suggest the need for some device to make the way we pick a path syntagmatically coherent.

The way we respond to the issue is by introducing explicit prohibitions, or 'drop-me-not' rules for POTs to comply with. Some of the major rules are shown in Table 1. A 'drop-me-not' rule (DMN) applies to a local dependency tree consisting of a parent node and its immediate child nodes. The intent of a DMN rule is to prohibit any one of the elements specified on the right hand side of the arrow from falling off in the presence of the head

node; they will be gone only if their head node is.

R1 says that if you have a dependency tree headed by VB with *nsubj*, *aux*, *neg*, or *mark* among its children, they should stay with VB; R2 prohibits against eliminating a WDT or WRB-labeled word in a dependency structure headed by VB; R6 disallows either *cc* or *conj* to drop without accompanying the other, for whatever type the head node assumes.

In Table 2, we find some examples that motivate the kinds of DMN rules we have in Table 1. Note that given the DMNs, the generation of '*was likely depend*,' or '*plan likely depend*' is no longer possible for the sentence in Figure 3.

6 Reranking with CRFs

Pipelining GST/g with CRFs allows us to tap into a host of features found in the sentence that could usefully be exploited toward generating compression, and requires no significant change in the way it is first conceived in Nomoto (2008), in order to make it work for English. It simply involves translating an output by GST/g into the form that allows the use of CRFs; this could be done simply by labeling words included in compression as '1' and those taken out as '0,' which would produce a binary representation of an output generated by GST/g. Given a source sentence $\mathbf{x}$ and a set $G(S)$ of candidate compressions generated by GST/g $-$ represented in binary format $-$ we seek to solve the following,

$$\mathbf{y}^{\star} = \underset{\mathbf{y} \in G(S)}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}). \tag{5}$$

where $\mathbf{y}^{\star}$ could be found using regular linear-chain CRFs (Lafferty et al., 2001). θ stands for model parameters. In building CRFs, we made use of features representing lexical forms, syntactic categories, dependency relations, TFIDF, whether a given word appears in the title of an article, and the left and right lexical contexts of a word.

7 T3

Cohn and Lapata (2008; 2009) are a recent attempt to bring a machine learning framework known as 'Structured SVM' to bear on sentence compression and could be considered to be among the current state-of-art approaches. Roughly speaking, their approach or what they call 'Tree-to-Tree Transducer' (T3) takes sentence compression to be the problem of classifying the source sentence with its target sentence, where one seeks to find some label $\mathbf{y}$, which represents a compression, for a given source sentence $\mathbf{x}$, that satisfies the following equation,

$$f(\mathbf{x};\mathbf{w}) = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}}\, F(\mathbf{y},\mathbf{x};\mathbf{w}), \tag{6}$$

and

$$F(\mathbf{y},\mathbf{x};\mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{y},\mathbf{x}) \rangle, \tag{7}$$

where $\mathbf{w}$, a vector representing model parameters, is determined in such a way that for a target class $\mathbf{y}$ and a prediction $\mathbf{y}'$, $F(\mathbf{x},\mathbf{y};\mathbf{w}) - F(\mathbf{x},\mathbf{y}';\mathbf{w}) > \Delta(\mathbf{y},\mathbf{y}') - \xi$, $\forall \mathbf{y}' \neq \mathbf{y}$; $\Delta(\mathbf{y},\mathbf{y}')$ represents a loss function and ξ a slack variable (Tsochantaridis et al., 2005). $\Psi(\mathbf{y},\mathbf{x})$ represents a vector of features culled from $\mathbf{y}$ and $\mathbf{x}$, and $\langle \cdot,\cdot \rangle$ a dot product.

For each of the rules used to derive a source sentence, T3 makes a decision on how or whether to transform the rule, with reference to $\langle \cdot,\cdot \rangle$, which takes into account such features as the number of terminals, root category, and lengths of frontiers, which eventually leads to a compression via a chart style dynamic programming.

Table 3: RSS item and its source

R *Two bombings rocked Iraq today, killing at least 80 in attacks at a shrine in Karbala and a police recruiting station in Ramadi.*

S *Baghdad, Jan. 5 – Two new suicide bombings rocked Iraq today, killing at least 80 in an attack at a shrine in the Shiite city of Karbala and a police recruiting station in the Sunni city of Ramadi.*

8 Corpus

Parting ways with previous work on sentence compression, which heavily relied on humans to create gold standard references, this work has a particular focus on using data gathered from RSS feeds, which if successful, could open a door to building gold standard data in large quantities rapidly and with little human effort. The primary objective of the present work is to come up with an approach capable of exploiting naturally occurring data as references for compression. So we are interested

Table 2: Examples. $a \Leftarrow_r b$ means that b stands in an r-relation to a.

rel, *nsubj*	In defying the President, Bill Frist was veering to the political center in a year during which he had artfully courted his party's right wing.	*couted* $\Leftarrow_{rel}$ *during* *veering* $\Leftarrow_{nsubj}$ *Bill Frist*
neg	Isaac B. Weisfuse says that the idea that a pandemic flu will somehow skip the 21st century does not make any sense.	*make* $\Leftarrow_{neg}$ *not*
mark	Prime Minister Ariel Sharon of Israel lashed out at protesters as troops finished clearing all but the last of the 21 Gaza settlements.	*finished* $\Leftarrow_{mark}$ *as*
WDT	The announcement offered few details that would convince protestants that they should resume sharing power with the I.R.A.'s political wing.	*convince* $\Leftarrow_{wdt}$ *that*
WRB	Arbitron, a company best known for its radio ratings, is testing a portable, pager-size device that tracks exposure to media throughout the day, wherever its wearer may go.	*go* $\Leftarrow_{wrb}$ *wherever*
cop	Buildings in a semi-abandoned town just inside Mexico that is a haven for would-be immigrants and smugglers will be leveled.	*haven* $\Leftarrow_{cop}$ *is*
aux, *poss*:WP	Harutoshi Fukui has penned a handful of best sellers whose common themes resonate in a country shedding its pacifism and rearming itself.	*resonate* $\Leftarrow_{poss:\text{WP}}$ *whose* *penned* $\Leftarrow_{aux}$ *has*

Table 4: RSS Corpus from NYTimes.com.

areas	# of items
INTERNATIONAL	2052
NYREGION	1153
NATIONAL	1351
OBITUARIES	541
OPINION	1031
SCIENCE	465
SPORTS	1451
TECHNOLOGY	978
WASHINGTON	1297

in finding out how GST compares with T3 from this particular perspective.

We gathered RSS feeds at NYTimes.com over a period of several months, across different sections, including INTERNATIONAL, NATIONAL, NYREGION, BUSINESS, and so forth, out of which we randomly chose 2,000 items for training data and 116 for testing data. For each RSS summary, we located its potential source sentence in the linked page, using a similarity metric known as SoftTFIDF (Cohen et al., 2003).[2] Table 4 gives a rundown on areas items came from and how many of them we collected for each of these areas.

For the ease of reference, we refer to a corpus of the training and test data combined as 'NYT-RSS,' and let 'NYT-RSS(A)' denote the training part of NYT-RSS, and 'NYT-RSS(B)' the testing part.

[2]SoftTFIDF is a hybrid of the TFIDF scheme and an edit-distance model known as Jaro-Winkler(Cohen et al., 2003).

9 Experiments

We ran the Stanford Parser on NYT-RSS to extract dependency structures for sentences involved, to be used with GST/g (de Marneffe et al., 2006; Klein and Manning, 2003). We manually developed 28 DMN rules out of NYT-RSS(A), some of which are presented in Table 1. An alignment between the source sentence and its corresponding gold standard compression was made by SWA or a standard sequence alignment algorithm by Smith and Waterman (1981). Importantly, we set up GST/g and T3 in such a way that they rely on the same set of dependency analyses and alignments when they are put into operation. We trained T3 on NYT-RSS(A) with default settings except for "–epsilon" and "–delete" options which we turned off, as preliminary runs indicated that their use led to a degraded performance (Cohn, 2008). We also set the loss function as was given in the default settings. We trained both GST/r, and T3 on NYT-RSS(A).

We ran GST/g and GST/g+r, i.e., GST/r pipelined with GST/g, varying the compression rate from 0.4 to 0.7. This involved letting GST/g rank candidate compressions by $S(\mathbf{p})$ and then choosing the first candidate to satisfy a given compression rate, whereas GST/g+r was made to output the highest ranking candidate as measured by $p(\mathbf{y} \mid \mathbf{x}; \theta)$, which meets a particular compression rate. It should be emphasized, however, that in T3, varying compression rate is not something the user has control over; so we accepted whatever output

Table 5: Results on NYT-RSS. '*'-marked figures mean that performance of GST/g is different from that of GST/g+r (on the comparable CompR) at 5% significance level according to t-test. The figures indicate average ratings.

Model	CompR	Intelligibility	Rep.
GST/g+r	0.446	2.836	2.612
GST/g	0.469	3.095	2.569
GST/g+r	0.540	2.957	2.767
GST/g	0.562	3.069	3.026*
GST/g+r	0.632	2.931	2.957
GST/g	0.651	3.060	3.259*
GST/g+r	0.729	3.155	3.345
GST/g	0.743	3.328	3.621*
T3	0.353	1.750	1.586
Gold Std.	0.657	4.776	3.931

T3 generated for a given sentence.

Table 5 shows how GST/g, GST/g+r, and T3 performed on NYT-RSS, along with the gold standard, on a scale of 1 to 5. Ratings were solicited from 4 native speakers of English. 'CompR' indicates compression rate. 'Intelligibility' means how well the compression reads; 'representativeness' how well the compression represents its source sentence. Table 6 presents a guideline for rating, describing what each rating should mean, which was also presented to human judges to facilitate evaluation.

The results in Table 5 indicate a clear superiority of GST/g and GST/g+r over T3, while differences in intelligibility between GST/g and GST/g+r were found not statistically significant. What is intriguing, though, is that GST/g produced performance statistically different in representativeness from GST/g+r at 5% level as marked by the asterisk.

Shown in Table 8 are examples of compression created by GST/g+r, GST/g and T3, together with gold standard compressions and relevant source sentences. One thing worth noting about the examples is that T3 keeps inserting out-of-the-source information into compression, which obviously has done more harm than good to compression.

Table 6: Guideline for Rating

MEANING	EXPLANATION	SCORE
very bad	For intelligibility, it means that the sentence in question is rubbish; no sense can be made out of it. As for representativeness, it means that there is no way in which the compression could be viewed as representing its source.	1
poor	Either the sentence is broken or fails to make sense for the most part, or it is focusing on points of least significance in the source.	2
fair	The sentence can be understood, though with some mental effort; it covers some of the important points in the source sentence.	3
good	The sentence allows easy comprehension; it covers most of important points talked about in the source sentence.	4
excellent	The sentence reads as if it were written by human; it gives a very good idea of what is being discussed in the source sentence.	5

Table 7: Examples from corpora. 'C' stands for reference compression; 'S' source sentence.

NYT-RSS

C *Jeanine F. Pirro said that she would abandon her plans to unseat senator Hillary Rodham Clinton and would instead run for state attorney general .*

S *Jeanine F. Pirro, whose campaign to unseat United States senator Hillary Rodham Clinton was in upheaval almost from the start, said yesterday that she would abandon the race and would instead run for attorney general of New York.*

CLwritten

C *Montserrat, the Caribbean island, is bracing itself for arrests following a fraud investigation by Scotland Yard.*

S *Montserrat, the tiny Caribbean island that once boasted one bank for every 40 inhabitants, is bracing itself this Easter for a spate of arrests following a three-year fraud and corruption investigation by Scotland Yard.*

CLspoken

C *This gives you the science behind the news, with topics explained in detail, from Mad Cow disease to comets.*

S *This page gives you the science behind the news, with hundreds of topics explained in detail, from Mad Cow disease to comets.*

Table 8: form GST/g+r, GST/g, T3, and Gold standard. ('Source' represents a source sentence.)

GST/g+r	The Corporation plans to announce today at the Game Show that it will begin selling the Xbox 360, its new video console , on Nov 22.
GST/g	The Microsoft Corporation plans to announce at the Tokyo Game Show that it will begin selling Xbox 360, new video console , on Nov.
T3	The Microsoft Corporation in New York plans to announce today at the Tokyo Game Show it will begin selling the Xbox 360 , its new video game console, on Nov 22.
Gold	The Microsoft Corporation plans to announce Thursday at the Tokyo Game Show that it will begin selling the Xbox 360 , its new video game console, on Nov. 22.
Source	The Microsoft Corporation plans to announce today at the Tokyo Game Show that it will begin selling the Xbox 360, its new video game console, on Nov 22.
GST/g+r	Scientists may have solved the chemical riddle of why the SARS virus causes such pneumonia and have developed a simple therapy.
GST/g	Scientists may have solved the chemical riddle of why the virus causes such a pneumonia and have developed a simple therapy.
T3	The scientists may solved the chemical riddle of the black river of why the SARS virus causes such a deadly pneumonia.
Gold	Scientists may have solved the riddle of why the SARS virus causes such a deadly pneumonia.
Source	Scientists may have solved the chemical riddle of why the SARS virus causes such a deadly pneumonia and have developed a simple therapy that promises to decrease the extraordinarily high death rate from the disease, according to a report in the issue of the journal nature-medicine that came out this week.
GST/g+r	A flu shot from GlaxoSmithKline was approved by American regulators and the Corporation vaccine plant, shut year because of, moved closer to being opened work to avoid.
GST/g	A flu shot was approved by regulators yesterday and the Chiron Corporation vaccine plant, shut , moved closer to being opened as officials work to avoid shortage.
T3	A flu shot from gaza was the Chiron Corporation's Liverpool vaccine plant, shut last year of a contamination shortage,, but critics suggest he is making it worse.
Gold	The Chiron Corporation's liverpool vaccine plant , shut last year because of contamination, moved closer to being opened as officials work to avoid another shortage.
Source	A flu shot from GlaxoSmithKline was approved by American regulators yesterday and the Chiron Corporation's Liverpool vaccine plant , shut last year because of contamination, moved closer to being opened as officials work to avoid another shortage.

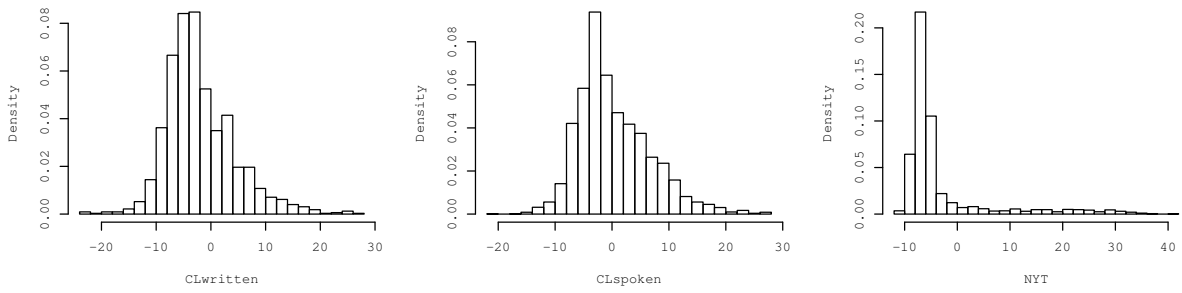


Figure 4: Density distribution of alignment scores. The x-dimension represents the degree of alignment between gold standard compression and its source sentence.

Table 9: Alignment Scores by SWA

NYT-RSS		CLwritten		CLspoken	
-3.061	(2000)	-1.882	(1629)	0.450	(4110)

10 Why T3 fails

It is interesting and worthwhile to ask what caused T3, heavily clad in ideas from the recent machine learning literature, to fail on NYT-RSS, as opposed to the 'CLwritten' and 'CLspoken' corpora on which T3 reportedly prevailed compared to other approaches (Cohn and Lapata, 2009). The CLwritten corpus comes from written sources in the British National Corpus and the American News Text corpus; the CLspoken corpus comes from transcribed broadcast news stories (cf. Table 7).

We argue that there are some important differences between the NYT-RSS corpus and the CLwritten/CLspoken corpora that may have led to T3's poor record with the former corpus.

The CLwritten and CLspoken corpora were created with a specific purpose in mind: namely to assess the compression-by-deletion approach. So their authors had a very good reason to limit gold standard compressions to those that can be arrived at only through deletion; annotators were carefully instructed to create compression by deleting words from the source sentence in a way that preserves the gist of the original sentence. By contrast, NYT-RSS consists of naturally occurring compressions sampled from live feeds on the Internet, where relations between compression and its source sentence are often not as straightforward. For instance, to arrive at a compression in NYT-RSS in Table 7 involves replacing *race* with *her plans to unseat senator Hillary Rodam Clinton*, which is obviously beyond what is possible with the deletion based approach.

In CLwritten and CLspoken, on the other hand, compressions are constructed out of parts that appear *in verbatim* in the original sentence, as Table 7 shows: thus one may get to the compressions by simply crossing off words from the original sentence.

To see whether there is any significant difference among NYT-RSS, CLwritten and CLspoken, we examined how well gold standard compressions are aligned with source sentences on each of the corpora, using SWA. Table 9 shows what we found. Parenthetical numbers represent how many pairs of compression and source are found in each corpus. A larger score means a tighter alignment between gold standard compression and its source sentence: we find in Table 9 that CLspoken has a source sentence more closely aligned with its compression than CLwritten, whose alignments are more closely tied than NYT-RSS's.

Figure 4 (found in the previous page) shows how SWA alignment scores are distributed over each of the corpora. CLwritten and CLspoken have peaks at around 0, with an almost entire mass of scores concentrating in an area close to or above 0. This means that for the most of the cases in either CLwritten or CLspoken, compression is very similar in form to its source. In contrast, NYT-RSS has a heavy concentration of scores in a stretch between -5 and -10, indicating that for the most of time, the overlap between compression and its source is rather modest compared to CLwritten and CLspoken.

So why does T3 fails on NYT-RSS? Because NYT-RSS contains lots of alignments that are only weakly related: in order for T3 to perform well, the training corpus should be made as free of spurious data as possible, so that most of the alignments are rated over and around 0 by SWA. Our concern is that such data may not happen naturally, as the density distribution of NYT-RSS shows, where the majority of alignments are found far below 0, which could raise some questions about the robustness of T3.

11 Conclusions

This paper introduced the model free approach, GST/g, which works by creating compressions only in reference to dependency structure, and looked at how it compares with a model intensive approach T3 on the data gathered from the Internet. It was found that the latter approach appears to crucially rely on the way the corpus is constructed in order for it to work, which may mean a huge compromise.

Interestingly enough, GST/g came out a winner on the particular corpus we used, even outperform-

ing its CRFs harnessed version, GST/g+r in representativeness. This suggests that we might gain more by improving fluency of GST/g than by focusing on its representativeness, which in any case came close to that of human at 70% compression level. The future work should also look at how the present approach fares on CLwritten and CLspoken, for which T3 was found to be effective.

Acknowledgements

The author likes to express gratitude to the reviewers of EMNLP for the time and trouble they took to review the paper. Their efforts are greatly appreciated.

References

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st COLING and 44th ACL*, pages 377–384, Sydney, Australia, July.

William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In Subbarao Kambhampati and Craig A. Knoblock, editors, *IIWeb*, pages 73–78.

Trevor Cohn and Mirella Lapata. 2007. Large margin synchronous generation and its application to sentence compression. In *Proceedings of the 2007 EMNLP-CoNLL*, pages 73–82, Prague, Czech Republic, June.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd COLING*, pages 137–144, Manchester, UK, August.

Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. Draft at http://homepages.inf.ed.ac.uk/tcohn/t3/.

Trevor Cohn. 2008. T3: Tree Transducer Toolkit. http://homepages.inf.ed.ac.uk/tcohn/t3/.

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

Katja Filippova and Michael Strube. 2007. Generating constituent order in german clauses. In *Proceedings of the 45th ACL*, pages 320–327, Prague, Czech Republic, June.

Katja Filippova and Michael Strube. 2008. Sentence fusion via dependency graph compression. In *Proceedings of the 2008 EMNLP*, pages 177–185, Honolulu, Hawaii, October.

C. Hori and Sadaoki Furui. 2004. Speech summarization: an approach through word extraction and a method for evaluation. *IEICE Transactions on Information and Systems*, E87-D(1):15–25.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st ACL*, pages 423–430, Sapporo, Japan, July.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139:91–107.

John Lafferty, Andrew MacCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th ICML-2001*.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of the 11th EACL*, pages 297–304.

Masaaki Nagata. 1994. A stochastic japanese morphological analyzer using a forward-dp backward-a* n-best search algorithm. In *Proceedings of COLING-94*.

Tadashi Nomoto. 2008. A generic sentence trimmer with CRFs. In *Proceedings of ACL-08: HLT*, pages 299–307, Columbus, Ohio, June.

Stefan Riezler, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical functional grammar. In *Proceedings of HLT-NAACL 2003*, pages 118–125, Edmonton.

T. F. Smith and M. S. Waterman. 1981. Identification of common molecular subsequence. *Journal of Molecular Biology*, 147:195–197.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2005. Support vector machine learning for interdependent and structured output spaces. *Journal of Machine Learning Research*, 6:1453–1484.

Jenie Turner and Eugen Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd ACL*, pages 290–297, Ann Arbor, June.

Kiwamu Yamagata, Satoshi Fukutomi, Kazuyuki Takagi, and Kzauhiko Ozeki. 2006. Sentence compression using statistical information about dependency path length. In *Proceedings of TSD 2006 (Lecture Notes in Computer Science, Vol. 4188/2006)*, pages 127–134, Brno, Czech Republic.

Natural Language Generation with Tree Conditional Random Fields

Wei Lu[1], **Hwee Tou Ng**[1,2], **Wee Sun Lee**[1,2]
[1]Singapore-MIT Alliance
[2]Department of Computer Science
National University of Singapore
luwei@nus.edu.sg
{nght,leews}@comp.nus.edu.sg

Abstract

This paper presents an effective method for generating natural language sentences from their underlying meaning representations. The method is built on top of a *hybrid tree* representation that jointly encodes both the meaning representation as well as the natural language in a tree structure. By using a tree conditional random field on top of the hybrid tree representation, we are able to explicitly model phrase-level dependencies amongst neighboring natural language phrases and meaning representation components in a simple and natural way. We show that the additional dependencies captured by the tree conditional random field allows it to perform better than directly inverting a previously developed hybrid tree semantic parser. Furthermore, we demonstrate that the model performs better than a previous state-of-the-art natural language generation model. Experiments are performed on two benchmark corpora with standard automatic evaluation metrics.

1 Introduction

One of the ultimate goals in the field of natural language processing (NLP) is to enable computers to converse with humans through human languages. To achieve this goal, two important issues need to be studied. First, it is important for computers to capture the meaning of a natural language sentence in a meaning representation. Second, computers should be able to produce a human-understandable natural language sentence from its meaning representation. These two tasks are referred to as semantic parsing and natural language generation (NLG), respectively.

In this paper, we use corpus-based statistical methods for constructing a natural language generation system. Given a set of pairs, where each pair consists of a natural language (NL) sentence and its formal meaning representation (MR), a learning method induces an algorithm that can be used for performing language generation from other previously unseen meaning representations.

A crucial question in any natural language processing system is the representation used. Meaning representations can be in the form of a tree structure. In Lu et al. (2008), we introduced a *hybrid tree* framework together with a probabilistic generative model to tackle semantic parsing, where tree structured meaning representations are used. The hybrid tree gives a natural joint tree representation of a natural language sentence and its meaning representation.

A joint generative model for natural language and its meaning representation, such as that used in Lu et al. (2008) has several advantages over various previous approaches designed for semantic parsing. First, unlike most previous approaches, the generative approach models a simultaneous generation process for both NL and MR. One elegant property of such a joint generative model is that it allows the modeling of both semantic parsing and natural language generation within the same process. Second, the generative process proceeds as a recursive top-down Markov process in a way that takes advantage of the tree structure of the MR. The hybrid tree generative model proposed in Lu et al. (2008) was shown to give state-of-the-art accuracy in semantic parsing on benchmark corpora.

While semantic parsing with hybrid trees has been studied in Lu et al. (2008), its inverse task – NLG with hybrid trees – has not yet been explored. We believe that the properties that make the hybrid trees effective for semantic parsing also make them effective for NLG. In this paper, we develop systems for the generation task by building

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 400–409,
Singapore, 6-7 August 2009.

on top of the generative model introduced in Lu et al. (2008) (referred to as the LNLZ08 system).

We first present a baseline model by directly "inverting" the LNLZ08 system, where an NL sentence is generated word by word. We call this model the *direct inversion model*. This model is unable to model some long range global dependencies over the entire NL sentence to be generated. To tackle several weaknesses exhibited by the baseline model, we next introduce an alternative, novel model that performs generation at the phrase level. Motivated by conditional random fields (CRF) (Lafferty et al., 2001), a different parameterization of the conditional probability of the hybrid tree that enables the model to encode some longer range dependencies amongst phrases and MRs is used. This novel model is referred to as the *tree* CRF*-based model*.

Evaluation results for both models are presented, through which we demonstrate that the tree CRF-based model performs better than the direct inversion model. We also compare the tree CRF-based model against the previous state-of-the-art model of Wong and Mooney (2007). Furthermore, we evaluate our model on a dataset annotated with several natural languages other than English (Japanese, Spanish, and Turkish). Evaluation results show that our proposed tree CRF-based model outperforms the previous model.

2 Related Work

There have been substantial earlier research efforts on investigating methods for transforming MR to their corresponding NL sentences. Most of the recent systems tackled the problem through the architecture of chart generation introduced by Kay (1996). Examples of such systems include the chart generator for Head-Driven Phrase Structure Grammar (HPSG) (Carroll et al., 1999; Carroll and Oepen, 2005; Nakanishi et al., 2005), and more recently for Combinatory Categorial Grammar (CCG) (White and Baldridge, 2003; White, 2004). However, most of these systems only focused on surface realization (inflection and ordering of NL words) and ignored lexical selection (learning the mappings from MR domain concepts to NL words).

The recent work by Wong and Mooney (2007) explored methods for generation by inverting a system originally designed for semantic parsing. They introduced a system named WASP^{-1} that employed techniques from statistical machine translation using Synchronous Context-Free Grammar (SCFG) (Aho and Ullman, 1972). The system took in a linearized MR tree as input, and translated it into a natural language sentence as output. Unlike most previous systems, their system integrated both lexical selection and surface realization in a single framework. The performance of the system was enhanced by incorporating models borrowed from PHARAOH (Koehn, 2004). Experiments show that this new hybrid system named WASP^{-1}++ gives state-of-the-art accuracies and outperforms the direct translation model obtained from PHARAOH, when evaluated on two corpora. We will compare our system's performance against that of WASP^{-1}++ in Section 5.

3 The Hybrid Tree Framework and the LNLZ08 System

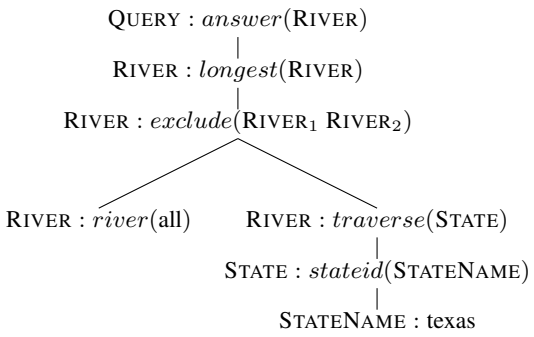


Figure 1: An example MR paired with its NL sentence.

Following most previous works in this area (Kate et al., 2005; Ge and Mooney, 2005; Kate and Mooney, 2006; Wong and Mooney, 2006; Lu et al., 2008), we consider MRs in the form of tree structures. An example MR and its corresponding natural language sentence are shown in Figure 1. The MR is a tree consisting of nodes called MR productions. For example, the node "QUERY : $answer$(RIVER)" is one MR production. Each MR production consists of a semantic category ("QUERY"), a function symbol ("$answer$") which can be optionally omitted, as well as an argument list which possibly contains

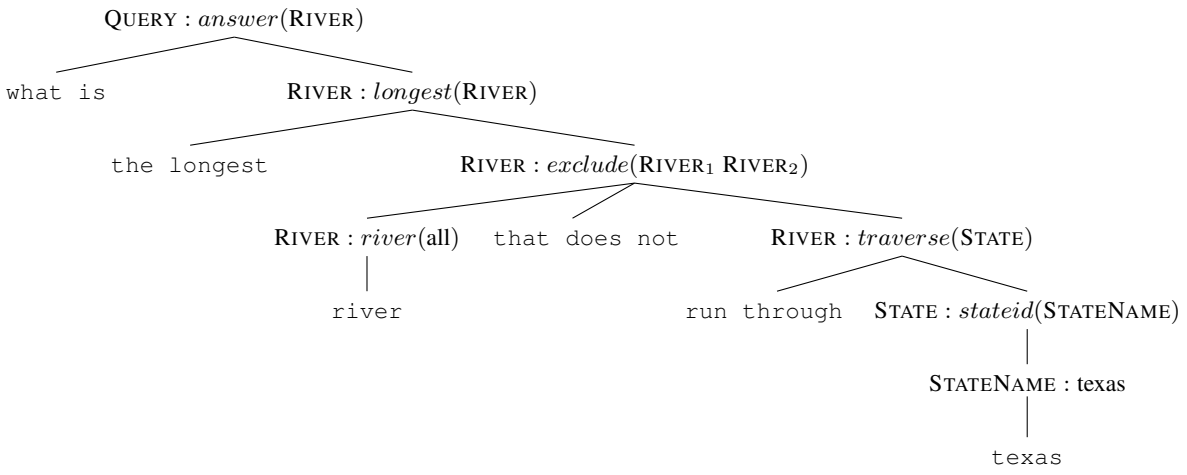


Figure 2: One possible hybrid tree $\mathcal{T}_1$

child semantic categories ("RIVER").

Now we give a brief overview of the hybrid tree framework and the LNLZ08 system that was presented in Lu et al. (2008). The training corpus required by the LNLZ08 system contains example pairs $\mathbf{d}^{(i)} = (\widehat{\mathbf{m}}^{(i)}, \widehat{\mathbf{w}}^{(i)})$ for $i = 1 \ldots N$, where each $\widehat{\mathbf{m}}^{(i)}$ is an MR, and each $\widehat{\mathbf{w}}^{(i)}$ is an NL sentence. The system makes the assumption that the entire training corpus is generated from an underlying generative model, which is specified by the parameter set Ω.

The parameter set Ω includes the following: the MR model parameter $\rho(m_j|m_i, \arg_k)$ which models the generation of an MR production m_j from its parent MR production m_i as its k-th child, the emission parameter $\theta(t|m_i, \Lambda)$ that is responsible for generation of an NL word or a semantic category t from the MR production m_i (the parent of t) under the context Λ (such as the token to the left of the current token), and the pattern parameter $\phi(r|m_i)$, which models the selection of a *hybrid pattern* r that defines globally how the NL words and semantic categories are interleaved given a parent MR production m_i. All these parameters are estimated from the corpus during the training phase. The list of possible hybrid patterns is given in Table 1 (at most two child semantic categories are allowed – MR productions with more child semantic categories are transformed into those with two).

In the table, m refers to the MR production, the symbol **w** denotes an NL word sequence and is optional if it appears inside []. The symbol $\mathcal{Y}$ and $\mathcal{Z}$ refer to the first and second semantic category under the MR production m respectively.

# RHS	Hybrid Pattern	# Patterns
0	$m \rightarrow$ **w**	1
1	$m \rightarrow$ [**w**]$\mathcal{Y}$[**w**]	4
2	$m \rightarrow$ [**w**]$\mathcal{Y}$[**w**]$\mathcal{Z}$[**w**]	8
	$m \rightarrow$ [**w**]$\mathcal{Z}$[**w**]$\mathcal{Y}$[**w**]	8

Table 1: The list of possible hybrid patterns, [] denotes optional

The generative process recursively creates MR productions as well as NL words at each generation step in a top-down manner. This process results in a *hybrid tree* for each MR-NL pair. The list of children under each MR production in the hybrid tree forms a *hybrid sequence*. One example hybrid tree for the MR-NL pair given in Figure 1 is shown in Figure 2. In this hybrid tree $\mathcal{T}_1$, the list of children under the production RIVER : $longest$(RIVER) forms the hybrid sequence "`the longest` RIVER : $exclude$(RIVER$_1$ RIVER$_2$)". The yield of the hybrid tree is exactly the NL sentence. The MR can also be recovered from the hybrid tree by recording all the internal nodes of the tree, subject to the reordering operation required by the hybrid pattern.

To illustrate, consider the generation of the hybrid tree $\mathcal{T}_1$ shown in Figure 2. The model first generates an MR production from its parent MR production (empty as the MR production is the root in the MR). Next, it selects a hybrid pattern $m \rightarrow \mathbf{w}\mathcal{Y}$ from the predefined list of hybrid patterns, which puts a constraint on the set of all allowable hybrid sequences that can be generated: the hybrid sequence must be an NL word sequence

followed by a semantic category. Finally, actual NL words and semantic categories are generated from the parent MR production. Now the generation for one level is complete, and the above process repeats at the newly generated MR productions, until the complete NL sentence and MR are both generated.

Mathematically, the above generative process yields the following formula that models the joint probability for the MR-NL pair, assuming the context Λ for the emission parameter is the preceding word or semantic category (*i.e.*, the bigram model is assumed, as discussed in Lu et al. (2008)):

$$\begin{aligned} & p\big(\mathcal{T}_1(\widehat{\mathbf{w}}, \widehat{\mathbf{m}})\big) \\ &= \rho(\textsc{Query} : answer(\textsc{River}) | -, \arg_1) \\ &\times \phi(m \rightarrow \mathbf{w}\mathcal{Y} | \textsc{Query} : answer(\textsc{River})) \\ &\times \theta(\texttt{what} | \textsc{Query} : answer(\textsc{River}), \texttt{BEGIN}) \\ &\times \theta(\texttt{is} | \textsc{Query} : answer(\textsc{River}), \texttt{what}) \\ &\times \theta(\textsc{River} | \textsc{Query} : answer(\textsc{River}), \texttt{is}) \\ &\times \theta(\texttt{END} | \textsc{Query} : answer(\textsc{River}), \textsc{River}) \\ &\times \rho(\textsc{River} : longest(\textsc{River}) | \\ &\textsc{Query} : answer(\textsc{River}), \arg_1) \times \ldots \end{aligned} \quad (1)$$

where $\mathcal{T}_1(\widehat{\mathbf{w}}, \widehat{\mathbf{m}})$ denotes the hybrid tree $\mathcal{T}_1$ which contains the NL sentence $\widehat{\mathbf{w}}$ and MR $\widehat{\mathbf{m}}$.

For each MR-NL pair in the training set, there can be potentially many possible hybrid trees associated with the pair. However, the correct hybrid tree is completely unknown during training. The correct hybrid tree is therefore treated as a hidden variable. An efficient inside-outside style algorithm (Baker, 1979) coupled with further dynamic programming techniques is used for efficient parameter estimation.

During the testing phase, the system makes use of the learned model parameters to determine the most probable hybrid tree given a new natural language sentence. The MR contained in that hybrid tree is the output of the system. Dynamic programming techniques similar to those of training are also employed for efficient decoding.

The generative model used in the LNLZ08 system has a natural symmetry, allowing for easy transformation from NL to MR, as well as from MR to NL. This provides the starting point for our work in "inverting" the LNLZ08 system to generate natural language sentences from the underlying meaning representations.

4 Generation with Hybrid Trees

The task of generating NL sentences from MRs can be defined as follows. Given a training corpus consisting of MRs paired with their NL sentences, one needs to develop algorithms that learn how to effectively "paraphrase" MRs with natural language sentences. During testing, the system should be able to output the most probable NL "paraphrase" for a given new MR.

The LNLZ08 system models $p(\mathcal{T}(\widehat{\mathbf{w}}, \widehat{\mathbf{m}}))$, the joint generative process for the hybrid tree containing both NL and MR. This term can be rewritten in the following way:

$$p(\mathcal{T}(\widehat{\mathbf{w}}, \widehat{\mathbf{m}})) = p(\widehat{\mathbf{m}}) \times p\left(\mathcal{T}(\widehat{\mathbf{w}}, \widehat{\mathbf{m}}) | \widehat{\mathbf{m}}\right) \quad (2)$$

In other words, we reach an alternative view of the joint generative process as follows. We choose to generate the complete MR $\widehat{\mathbf{m}}$ first. Given $\widehat{\mathbf{m}}$, we generate hybrid sequences below each of its MR production, which gives us a complete hybrid tree $\mathcal{T}(\widehat{\mathbf{w}}, \widehat{\mathbf{m}})$. The NL sentence $\widehat{\mathbf{w}}$ can be constructed from this hybrid tree exactly.

We define an operation $yield(\mathcal{T})$ which returns the NL sentence as the yield of the hybrid tree $\mathcal{T}$. Given an MR $\widehat{\mathbf{m}}$, we find the most probable NL sentence $\widehat{\mathbf{w}}^*$ as follows:

$$\widehat{\mathbf{w}}^* = yield\Big(\operatorname*{argmax}_{\mathcal{T}} p(\mathcal{T} | \widehat{\mathbf{m}})\Big) \quad (3)$$

In other words, we first find the most probable hybrid tree $\mathcal{T}$ that contains the provided MR $\widehat{\mathbf{m}}$. Next we return the yield of $\mathcal{T}$ as the most probable NL sentence.

Different assumptions can be made in the process of finding the most probable hybrid tree. We first describe a simple model which is a direct inversion of the LNLZ08 system. This model, as a baseline model, generates a complete NL sentence word by word. Next, a more sophisticated model that exploits NL phrase-level dependencies is built that tackles some weaknesses of the simple baseline model.

4.1 Direct Inversion Model

Assume that a pre-order traversal of the MR $\widehat{\mathbf{m}}$ gives us the list of MR productions $m_1, m_2, \ldots, m_S$, where S is the number of MR productions in $\widehat{\mathbf{m}}$. Based on the independence assumption made by the LNLZ08 system, each MR production independently generates a hybrid

sequence. Denote the hybrid sequence generated under the MR production m_s as h_s, for $s = 1, \ldots, S$. We call the list of hybrid sequences $\mathbf{h} = \langle h_1, h_2, \ldots, h_S \rangle$ a *hybrid sequence list* associated with this particular MR. Thus, our goal is to find the optimal hybrid sequence list $\mathbf{h}^*$ for the given MR $\widehat{\mathbf{m}}$, which is formulated as follows:

$$\mathbf{h}^* = \langle h_1^*, \ldots, h_S^* \rangle = \underset{h_1,\ldots,h_S}{\operatorname{argmax}} \prod_{s=1}^{S} p(h_s|m_s) \quad (4)$$

The optimal hybrid sequence list defines the optimal hybrid tree whose yield gives the optimal NL sentence.

Due to the strong independence assumption introduced by the LNLZ08 system, the hybrid tree generation process is in fact highly decomposable. Optimization of the hybrid sequence list $\langle h_1, \ldots, h_S \rangle$ can be performed individually since they are independent of one another. Thus, mathematically, for $s = 1, \ldots, S$, we have:

$$h_s^* = \underset{h_s}{\operatorname{argmax}}\, p(h_s|m_s) \quad (5)$$

The LNLZ08 system presented three models for the task of transforming NL to MR. In this inverse task, for generation of a hybrid sequence, we choose to use the bigram model (model II). We choose this model mainly due to its stronger ability in modeling dependencies between adjacent NL words, which we believe to be quite important in this NL generation task. With the bigram model assumption, the optimal hybrid sequence that can be generated from each MR production is defined as follows:

$$\begin{aligned} h_s^* &= \underset{h_s}{\operatorname{argmax}}\, p(h_s|m_s) \\ &= \underset{h_s}{\operatorname{argmax}} \left\{ \phi(r|m_s) \times \prod_{j=1}^{|h_s|+1} \theta(t_j|m_s, t_{j-1}) \right\} \end{aligned} \quad (6)$$

where t_i is either an NL word or a semantic category with $t_0 \equiv$ BEGIN and $t_{|h_s|+1} \equiv$ END, and r is the hybrid pattern that matches the hybrid sequence h_s, which is equivalent to $t_1, \ldots, t_{|h_s|}$.

Equivalently, we can view the problem in the log-space:

$$\begin{aligned} h_s^* = \underset{h_s}{\operatorname{argmin}} \Bigg\{ & -\log \phi(r|m_s) \\ & + \sum_{j=1}^{|h_s|+1} -\log \theta(t_j|m_s, t_{j-1}) \Bigg\} \end{aligned} \quad (7)$$

Note the term $-\log \phi(r|m_s)$ is a constant for a particular MR production m_s and a particular hybrid pattern r. This search problem can be equivalently cast as the shortest path problem which can be solved efficiently with Dijkstra's algorithm (Cormen et al., 2001). We define a set of states. Each state represents a single NL word or a semantic category, including the special symbols BEGIN and END. A directed path between two different states t_u and t_v is associated with a distance measure $-\log \theta(t_v|m_s, t_u)$, which is non-negative. The task now is to find the shortest path between BEGIN and END[1]. The sequence of words appearing in this path is simply the most probable hybrid sequence under this MR production m_s. We build this model by directly inverting the LNLZ08 system, and this model is therefore referred to as the *direct inversion model*.

A major weakness of this baseline model is that it encodes strong independence assumptions during the hybrid tree generation process. Though shown to be effective in the task of transforming NL to MR, such independence assumptions may introduce difficulties in this NLG task. For example, consider the MR shown in Figure 1. The generation steps of the hybrid sequences from the two adjacent MR productions QUERY : $answer$(RIVER) and RIVER : $longest$(RIVER) are completely independent of each other. This may harm the fluency of the generated NL sentence, especially when a transition from one hybrid sequence to another is required. In fact, due to such an independence assumption, the model always generates the same hybrid sequence from the same MR production, regardless of its context such as parent or child MR productions. Such a limitation points to the importance of better utilizing the tree structure of the MR for this NLG task. Furthermore, due to the bigram assumption, the model is unable to capture longer range dependencies amongst the words or semantic categories in each hybrid sequence.

To tackle the above issues, we explore ways of relaxing various assumptions, which leads to an

[1] In addition, we should make sure that the generated hybrid sequence $t_0 \ldots t_{|h_s|+1}$ is a valid hybrid sequence that comply with the hybrid pattern r. For example, the MR production STATE : loc_1(RIVER) can generate the following hybrid sequence "BEGIN have RIVER END" but not this hybrid sequence "BEGIN have END". This can be achieved by finding the shortest path from BEGIN to RIVER, which then gets concatenated to the shortest path from RIVER to END.

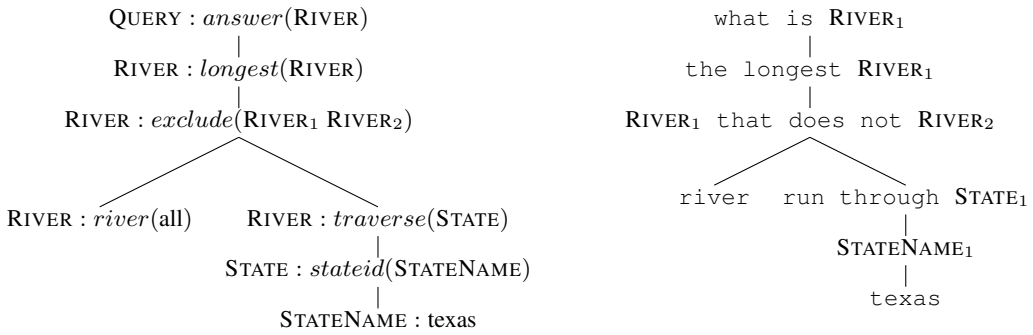


Figure 3: An MR (left) and its associated hybrid sequences (right)

alternative model as discussed next.

4.2 Tree CRF-Based Model

Based on the belief that using known phrases usually leads to better fluency in the NLG task (Wong and Mooney, 2007), we explore methods for generating an NL sentence at phrase level rather than at word level. This is done by generating hybrid sequences as complete objects, rather than sequentially one word or semantic category at a time, from MR productions.

We assume that each MR production can generate a complete hybrid sequence below it from a finite set of possible hybrid sequences. Each such hybrid sequence is called a *candidate hybrid sequence* associated with that particular MR production. Given a set of candidate hybrid sequences associated with each MR production, the generation task is to find the optimal hybrid sequence list $\mathbf{h}^*$ for a given MR $\widehat{\mathbf{m}}$:

$$\mathbf{h}^* = \underset{\mathbf{h}}{\operatorname{argmax}}\, p(\mathbf{h}|\widehat{\mathbf{m}}) \tag{8}$$

Figure 3 shows a complete MR, as well as a possible tree that contains hybrid sequences associated with the MR productions. For example, in the figure the MR production RIVER : $traverse$(STATE) is associated with the hybrid sequence `run through` STATE$_1$. Each MR production can be associated with potentially many different hybrid sequences. The task is to determine the most probable list of hybrid sequences as the ones appearing on the right of Figure 3, one for each MR production.

To make better use of the tree structure of MR, we take the approach of modeling the conditional distribution using a log-linear model. Following the conditional random fields (CRF) framework (Lafferty et al., 2001), we can define the probability of the hybrid sequence list given the complete MR $\widehat{\mathbf{m}}$, as follows:

$$p(\mathbf{h}|\widehat{\mathbf{m}}) = \frac{1}{Z(\widehat{\mathbf{m}})} \exp\left(\sum_{i \in V} \sum_{k} \mu_k g_k(h_i, \widehat{\mathbf{m}}, i) + \sum_{(i,j) \in E} \sum_{k} \lambda_k f_k(h_i, h_j, \widehat{\mathbf{m}}, i, j) \right) \tag{9}$$

where V is the set of all the vertices in the tree, and E is the set of the edges in the tree, consisting of parent-child pairs. The function $Z(\widehat{\mathbf{m}})$ is the normalization function. Note that the dependencies among the features here form a tree, unlike the sequence models used in Lafferty et al. (2001). The function $f_k(h_i, h_j, \widehat{\mathbf{m}}, i, j)$ is a feature function of the entire MR tree $\widehat{\mathbf{m}}$ and the hybrid sequences at vertex i and j. These features are usually referred to as the edge features in the CRF framework. The function $g_k(h_i, \widehat{\mathbf{m}}, i)$ is a feature function of the hybrid sequence at vertex i and the entire MR tree. These features are usually referred to as the vertex features. The parameters λ_k and μ_k are learned from the training data.

In this task, we are given only MR-NL pairs and do not have the hybrid tree corresponding to each MR as training data. Now we describe how the set of candidate hybrid sequences for each MR production is obtained as well as how the training data for this model is constructed. After the joint generative model is learned as done in Lu et al. (2008), we first use a Viterbi algorithm to find the optimal hybrid tree for each MR-NL pair in the training set. From each optimal hybrid tree, we extract the hybrid sequence h_i below each MR production m_i. Using this process on the training MR-NL pairs, we can obtain a set of candidate

hybrid sequences that can be associated with each MR production. The optimal hybrid tree generated by the Viterbi algorithm in this way is considered the "correct" hybrid tree for the MR-NL pair and is used as training data. While this does not provide hand-labeled training data, we believe the hybrid trees generated this way form a high quality training set as both the MR and NL are available when Viterbi decoding is performed, guaranteeing that the generated hybrid tree has the correct yield.

There exist several advantages of such a model over the simple generative model. First, this model allows features that specifically model the dependencies between neighboring hybrid sequences in the tree to be used. In addition, the model can efficiently capture long range dependencies between MR productions and hybrid sequences since each hybrid sequence is allowed to depend on the entire MR tree.

For features, we employ four types of simple features, as presented below. Note that the first three types of features are vertex features, and the last are edge features. Examples are given based on Figure 3. All the features are indicator functions, *i.e.*, a feature takes value 1 if a certain combination is present, and 0 otherwise. The last three features explicitly encode information from the tree structure of MR.

Hybrid sequence features : one hybrid sequence together with the associated MR production. For example:

$$g_1 : \langle \texttt{run through } \textsc{State}_1, \textsc{River} : traverse(\textsc{State}) \rangle ;$$

Two-level hybrid sequence features : one hybrid sequence, its associated MR production, and the parent MR production. For example:

$$g_2 : \langle \texttt{run through } \textsc{State}_1, \textsc{River} : traverse(\textsc{State}), \textsc{River} : exclude(\textsc{River}_1, \textsc{River}_2) \rangle ;$$

Three-level hybrid sequence features : one hybrid sequence, its associated MR production, the parent MR production, and the grandparent MR production. For example:

$$g_3 : \langle \texttt{run through } \textsc{State}_1, \textsc{River} : traverse(\textsc{State}), \textsc{River} : exclude(\textsc{River}_1, \textsc{River}_2), \textsc{River} : longest(\textsc{River}) \rangle ;$$

Adjacent hybrid sequence features : two adjacent hybrid sequences, together with their associated MR productions. For example:

$$f_1 : \langle \texttt{run through } \textsc{State}_1, \textsc{River}_1 \texttt{ that does not } \textsc{River}_2, \textsc{River} : traverse(\textsc{State}), \textsc{River} : exclude(\textsc{River}_1, \textsc{River}_2) \rangle .$$

For training, we use the feature forest model (Miyao and Tsujii, 2008), which was originally designed as an efficient algorithm for solving maximum entropy models for data with complex structures. The model enables efficient training over packed trees that potentially represent exponential number of trees. The tree conditional random fields model can be effectively represented using the feature forest model. The model has also been successfully applied to the HPSG parsing task.

To train the model, we run the Viterbi algorithm on the trained LNLZ08 model and perform convex optimization using the feature forest model. The LNLZ08 model is trained using an EM algorithm with time complexity $O(MN^3D)$ per EM iteration, where M and N are respectively the maximum number of MR productions and NL words for each MR-NL pair, and D is the number of training instances. The time complexity of the Viterbi algorithm is also $O(MN^3D)$. For training the feature forest, we use the Amis toolkit (Miyao and Tsujii, 2002) which utilizes the GIS algorithm. The time complexity for each iteration of the GIS algorithm is $O(MK^2D)$, where K is the maximum number of candidate hybrid sequences associated with each MR production. Finally, the time complexity for generating a natural language sentence from a particular MR is $O(MK^2)$.

5 Experiments

In this section, we present the results of our systems when evaluated on two standard benchmark corpora. The first corpus is GEOQUERY, which contains Prolog-based MRs that can be used to query a US geographic database (Kate et al., 2005). Our task for this domain is to generate NL sentences from the formal queries. The second corpus is ROBOCUP. This domain contains MRs which are instructions written in a formal language called CLANG. Our task for this domain is to generate NL sentences from the coaching advice written in CLANG.

	GEOQUERY (880)		ROBOCUP (300)	
	BLEU	NIST	BLEU	NIST
Direct inversion model	0.3973	5.5466	0.5468	6.6738
Tree CRF-based model	**0.5733**	**6.7459**	**0.6220**	**6.9845**

Table 2: Results of automatic evaluation of both models (**bold** type indicates the best performing system).

	GEOQUERY (880)		ROBOCUP (300)	
	BLEU	NIST	BLEU	NIST
WASP^{-1}++	0.5370	6.4808	0.6022	6.8976
Tree CRF-based model	**0.5733**	**6.7459**	**0.6220**	**6.9845**

Table 3: Results of automatic evaluation of our tree CRF-based model and WASP^{-1}++.

	English		*Japanese*		*Spanish*		*Turkish*	
	BLEU	NIST	BLEU	NIST	BLEU	NIST	BLEU	NIST
WASP^{-1}++	0.6035	5.7133	0.6585	4.6648	0.6175	5.7293	0.4824	4.3283
Tree CRF-based model	**0.6265**	**5.8907**	**0.6788**	**4.8486**	**0.6382**	**5.8488**	**0.5096**	**4.5033**

Table 4: Results on the GEOQUERY-250 corpus with 4 natural languages.

The GEOQUERY domain contains 880 instances, while the ROBOCUP domain contains 300 instances. The average NL sentence length for the two corpora are 7.57 and 22.52 respectively. Following the evaluation methodology of Wong and Mooney (2007), we performed 4 runs of the standard 10-fold cross validation and report the averaged performance in this section using the standard automatic evaluation metric BLEU (Papineni et al., 2002) and NIST (Doddington, 2002)[2]. The BLEU and NIST scores of the WASP^{-1}++ system reported in this section are obtained from the published paper of Wong and Mooney (2007). Note that to make our experimental results directly comparable to Wong and Mooney (2007), we used the identical training and test data splits for the 4 runs of 10-fold cross validation used by Wong and Mooney (2007) on both corpora.

Our system has the advantage of always producing an NL sentence given any input MR, even if there exist unseen MR productions in the input MR. We can achieve this by simply skipping those unseen MR productions during the generation process. However, in order to make a fair comparison against WASP^{-1}++, which can only generate NL sentences for 97% of the input MRs, we also do not generate any NL sentence in the case of observing an unseen MR production. All the evaluations discussed in this section follow this evaluation methodology, but we notice that empirically our system is able to achieve higher BLEU/NIST scores if we allow generation for those MRs that include unseen MR productions.

[2]We used the official evaluation script (version 11b) provided by http://www.nist.gov/.

5.1 Comparison between the two models

We compare the performance of our two models in Table 2. From the table, we observe that the tree CRF-based model outperforms the direct inversion model on both domains. This validates our earlier belief that some long range dependencies are important for the generation task. In addition, while the direct inversion model performs reasonably well on the ROBOCUP domain, it performs substantially worse on the GEOQUERY domain where the sentence length is shorter. We note that the evaluation metrics are strongly correlated with the cumulative matching n-grams between the output and the reference sentence (n ranges from 1 to 4 for BLEU, and 1 to 5 for NIST). The direct inversion model fails to capture the transitional behavior from one phrase to another, which makes it more vulnerable to n-gram mismatch, especially when evaluated on the GEOQUERY corpus where phrase-to-phrase transitions are more frequent. On the other hand, the tree CRF-based model does not suffer from this problem, mainly due to its ability to model such dependencies between neighboring phrases. Sample outputs from the two models are shown in Figure 4.

Reference:	`what is the largest state bordering texas`
Direct inversion model:	`what the largest states border texas`
Tree CRF-based model:	`what is the largest state that borders texas`
Reference:	`if DR2C7 is true then players 2 , 3 , 7 and 8 should pass to player 4`
Direct inversion model:	`if DR2C7 , then players 2 , 3 7 and 8 should ball to player 4`
Tree CRF-based model:	`if the condition DR2C7 is true then players 2 , 3 , 7 and 8 should pass to player 4`

Figure 4: Sample outputs from the two models, for GEOQUERY domain (top) and ROBOCUP domain (bottom) respectively.

5.2 Comparison with previous model

We also compare the performance of our tree CRF-based model against the previous state-of-the-art system WASP^{-1}++ in Table 3. Our tree CRF-based model achieves better performance on both corpora. We are unable to carry out statistical significance tests since the detailed BLEU and NIST scores of the cross validation runs of WASP^{-1}++ as reported in the published paper of Wong and Mooney (2007) are not available.

The results confirm our earlier discussions: the dependencies between the generated NL words are important and need to be properly modeled. The WASP^{-1}++ system uses a log-linear model which incorporates two major techniques to attempt to model such dependencies. First, a back-off language model is used to capture dependencies at adjacent word level. Second, a technique that merges smaller translation rules into a single rigid rule is used to capture dependencies at phrase level (Wong, 2007). In contrast, the proposed tree CRF-based model is able to explicitly and flexibly exploit phrase-level features that model dependencies between adjacent phrases. In fact, with the hybrid tree framework, the better treatment of the tree structure of MR enables us to model some crucial dependencies between the complete MR tree and generated NL phrases. We believe that this property plays an important role in improving the quality of the generated sentences in terms of fluency, which is assessed by the evaluation metrics.

Furthermore, WASP^{-1}++ employs minimum error rate training (Och, 2003) to directly optimize the evaluation metrics. We have not done so but still obtain better performance. In future, we plan to explore ways to directly optimize the evaluation metrics in our system.

5.3 Experiments on different languages

Following the work of Wong and Mooney (2007), we also evaluated our system's performance on a subset of the GEOQUERY corpus with 250 instances, where sentences of 4 natural languages (English, Japanese, Spanish, and Turkish) are available. The evaluation results are shown in Table 4. Our tree CRF-based model achieves better performance on this task compared to WASP^{-1}++. We are again unable to conduct statistical significance tests for the same reason reported earlier.

6 Conclusions

In this paper, we presented two novel models for the task of generating natural language sentences from given meaning representations, under a hybrid tree framework. We first built a simple direct inversion model as a baseline. Next, to address the limitations associated with the direct inversion model, a tree CRF-based model was introduced. We evaluated both models on standard benchmark corpora. Evaluation results show that the tree CRF-based model performs better than the direct inversion model, and that the tree CRF-based model also outperforms WASP^{-1}++, which was a previous state-of-the-art system reported in the literature.

Acknowledgments

The authors would like to thank Seung-Hoon Na for his suggestions on the presentation of this paper, Yuk Wah Wong for answering various questions related to the WASP^{-1}++ system, and the anonymous reviewers for their thoughtful comments on this work.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*. Prentice-Hall, Englewood Clis, NJ.

James K. Baker. 1979. Trainable grammars for speech recognition. In *Proceedings of the Spring Conference of the Acoustical Society of America*, pages 547–550, Boston, MA, June.

John Carroll and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP 2005)*, pages 165–176.

John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznanski. 1999. An efficient chart generator for (semi-) lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG 1999)*, pages 86–95.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. *Introduction to Algorithms (Second Edition)*. MIT Press.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the 2nd International Conference on Human Language Technology Research (HLT 2002)*, pages 138–145.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL 2005)*, pages 9–16.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006)*, pages 913–920.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 1062–1068.

Martin Kay. 1996. Chart generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL 1996)*, pages 200–204.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas (AMTA 2004)*, pages 115–124.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, pages 282–289.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 783–792.

Yusuke Miyao and Jun'ichi Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proceedings of the 2nd International Conference on Human Language Technology Research (HLT 2002)*, pages 292–297.

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

Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT 2005)*, volume 5, pages 93–102.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 160–167.

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

Michael White and Jason Baldridge. 2003. Adapting chart realization to CCG. In *Proceedings of the 9th European Workshop on Natural Language Generation (EWNLG 2003)*, pages 119–126.

Michael White. 2004. Reining in CCG chart realization. In *Proceeding of the 3rd International Conference on Natural Language Generation (INLG 2004)*, pages 182–191.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL 2006)*, pages 439–446.

Yuk Wah Wong and Raymond J. Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL/HLT 2007)*, pages 172–179.

Yuk Wah Wong. 2007. *Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques*. Ph.D. thesis, The University of Texas at Austin.

Perceptron Reranking for CCG Realization

Michael White and **Rajakrishnan Rajkumar**
Department of Linguistics
The Ohio State University
Columbus, OH, USA
{mwhite,raja}@ling.osu.edu

Abstract

This paper shows that discriminative reranking with an averaged perceptron model yields substantial improvements in realization quality with CCG. The paper confirms the utility of including language model log probabilities as features in the model, which prior work on discriminative training with log linear models for HPSG realization had called into question. The perceptron model allows the combination of multiple n-gram models to be optimized and then augmented with both syntactic features and discriminative n-gram features. The full model yields a state-of-the-art BLEU score of 0.8506 on Section 23 of the CCGbank, to our knowledge the best score reported to date using a reversible, corpus-engineered grammar.

1 Introduction

In this paper, we show how discriminative training with averaged perceptron models (Collins, 2002) can be used to substantially improve surface realization with Combinatory Categorial Grammar (Steedman, 2000, CCG). Velldal and Oepen (2005) and Nakanishi et al. (2005) have shown that discriminative training with log-linear (maximum entropy) models is effective in realization ranking with Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994, HPSG). Here we show that averaged perceptron models also perform well for realization ranking with CCG. Averaged perceptron models are very simple, just requiring a decoder and a simple update function, yet despite their simplicity they have been shown to achieve state-of-the-art results in Treebank and CCG parsing (Huang, 2008; Clark and Curran, 2007a) as well as on other NLP tasks.

Along the way, we address the question of whether it is beneficial to incorporate n-gram log probabilities as baseline features in a discriminatively trained realization ranking model. On a limited domain corpus, Velldal & Oepen found that including the n-gram log probability of each candidate realization as a feature in their log-linear model yielded a substantial boost in ranking performance; on the Penn Treebank (PTB), however, Nakanishi et al. found that including an n-gram log prob feature in their model was of no benefit (with the use of bigrams instead of 4-grams suggested as a possible explanation). With these mixed results, the utility of n-gram baseline features for PTB-scale discriminative realization ranking has been unclear. In our particular setting, the question is: Do n-gram log prob features improve performance in broad coverage realization ranking with CCG, where factored language models over words, part-of-speech tags and supertags have previously been employed (White et al., 2007; Espinosa et al., 2008)?

We answer this question in the affirmative, confirming the results of Velldal & Oepen, despite the differences in corpus size and kind of language model. We show that including n-gram log prob features in the perceptron model is highly beneficial, as the discriminative models we tested without these features performed worse than the generative baseline. These findings are in line with Collins & Roark's (2004) results with incremental parsing with perceptrons, where it is suggested that a generative baseline feature provides the perceptron algorithm with a much better starting point for learning. We also show that discriminative training allows the combination of multiple n-gram models to be optimized, and that the best model augments the n-gram log prob features with both syntactic features and discriminative n-gram features. The full model yields a state-of-the-art BLEU (Papineni et al., 2002) score of 0.8506 on Section 23 of the CCGbank, which is to our knowledge the best score reported to date

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 410–419,
Singapore, 6-7 August 2009.

using a reversible, corpus-engineered grammar.

The paper is organized as follows. Section 2 reviews previous work on broad coverage realization with OpenCCG. Section 3 describes our approach to realization reranking with averaged perceptron models. Section 4 presents our evaluation of the perceptron models, comparing the results of different feature sets. Section 5 compares our results to those obtained by related systems and discusses the difficulties of cross-system comparisons. Finally, Section 6 concludes with a summary and discussion of future directions for research.

2 Background

2.1 Surface Realization with CCG

CCG (Steedman, 2000) is a unification-based categorial grammar formalism which is defined almost entirely in terms of lexical entries that encode sub-categorization information as well as syntactic feature information (e.g. number and agreement). Complementing function application as the standard means of combining a head with its argument, type-raising and composition support transparent analyses for a wide range of phenomena, including right-node raising and long distance dependencies. An example syntactic derivation appears in Figure 1, with a long-distance dependency between *point* and *make*. Semantic composition happens in parallel with syntactic composition, which makes it attractive for generation.

OpenCCG is a parsing/generation library which works by combining lexical categories for words using CCG rules and multi-modal extensions on rules (Baldridge, 2002) to produce derivations. Surface realization is the process by which logical forms are transduced to strings. OpenCCG uses a hybrid symbolic-statistical chart realizer (White, 2006) which takes logical forms as input and produces sentences by using CCG combinators to combine signs. Edges are grouped into equivalence classes when they have the same syntactic category and cover the same parts of the input logical form. Alternative realizations are ranked using integrated n-gram or perceptron scoring, and pruning takes place within equivalence classes of edges. To more robustly support broad coverage surface realization, OpenCCG greedily assembles fragments in the event that the realizer fails to find a complete realization.

To illustrate the input to OpenCCG, consider the semantic dependency graph in Figure 2. In

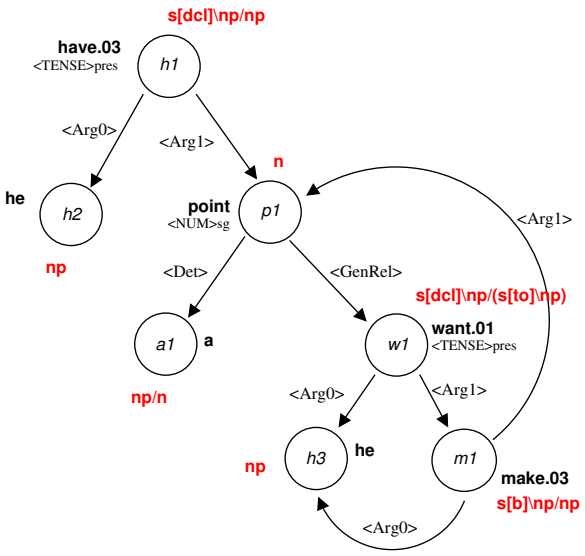


Figure 2: Semantic dependency graph from the CCGbank for *He has a point he wants to make [. . .]*, along with gold-standard supertags (category labels)

the graph, each node has a lexical predication (e.g. **make.03**) and a set of semantic features (e.g. ⟨NUM⟩sg); nodes are connected via dependency relations (e.g. ⟨ARG0⟩). (Gold-standard supertags, or category labels, are also shown; see Section 2.4 for their role in hypertagging.) Internally, such graphs are represented using Hybrid Logic Dependency Semantics (HLDS), a dependency-based approach to representing linguistic meaning (Baldridge and Kruijff, 2002). In HLDS, each semantic head (corresponding to a node in the graph) is associated with a nominal that identifies its discourse referent, and relations between heads and their dependents are modeled as modal relations.

2.2 Realization from an Enhanced CCGbank

Our starting point is an enhanced version of the CCGbank (Hockenmaier and Steedman, 2007)—a corpus of CCG derivations derived from the Penn Treebank—with Propbank (Palmer et al., 2005) roles projected onto it (Boxwell and White, 2008). To engineer a grammar from this corpus suitable for realization with OpenCCG, the derivations are first revised to reflect the lexicalized treatment of coordination and punctuation assumed by the multi-modal version of CCG that is implemented in OpenCCG (White and Rajkumar, 2008). Further changes are necessary to support semantic dependencies rather than surface syntactic ones; in

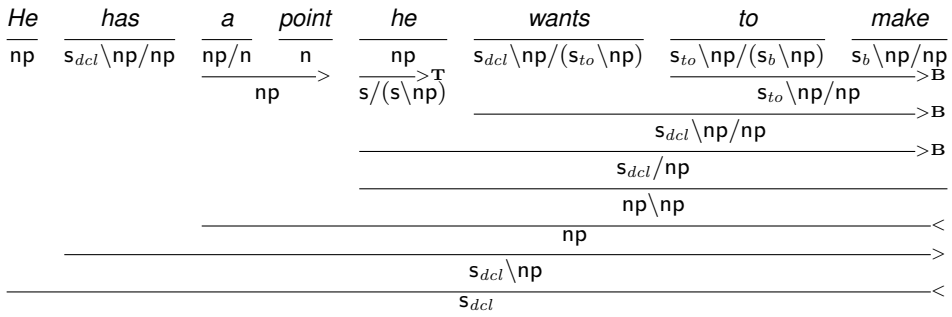


Figure 1: Syntactic derivation from the CCGbank for *He has a point he wants to make [. . .]*

particular, the features and unification constraints in the categories related to semantically empty function words such complementizers, infinitival-*to*, expletive subjects, and case-marking prepositions are adjusted to reflect their purely syntactic status.

In the second step, a grammar is extracted from the converted CCGbank and augmented with logical forms. Categories and unary type changing rules (corresponding to zero morphemes) are sorted by frequency and extracted if they meet the specified frequency thresholds. A separate transformation then uses a few dozen generalized templates to add logical forms to the categories, in a fashion reminiscent of (Bos, 2005). As shown in Figure 2, numbered semantic roles are taken from PropBank when available, and more specific relations are introduced in the categories for closed-class items such as determiners.

After logical form insertion, the extracted and augmented grammar is loaded and used to parse the sentences in the CCGbank according to the gold-standard derivation. If the derivation can be successfully followed, the parse yields a logical form which is saved along with the corpus sentence in order to later test the realizer. Currently, the algorithm succeeds in creating logical forms for 98.85% of the sentences in the development section (Sect. 00) of the converted CCGbank, and 97.06% of the sentences in the test section (Sect. 23). Of these, 95.99% of the development LFs are semantic dependency graphs with a single root, while 95.81% of the test LFs have a single root. The remaining cases, with multiple roots, are missing one or more dependencies required to form a fully connected graph. Such missing dependencies usually reflect remaining inadequacies in the logical form templates.

An error analysis of OpenCCG output by Rajkumar et al. (2009) recently revealed that out of 2331 named entities (NEs) annotated by the BBN corpus (Weischedel and Brunstein, 2005), 238 were not realized correctly. For example, multi-word NPs like *Texas Instruments Japan Ltd.* were realized as *Japan Texas Instruments Ltd.* Accordingly, inspired by Hogan et al.'s (2007)'s Experiment 1, Rajkumar et al. used the BBN corpus NE annotation to collapse certain classes of NEs. But unlike Hogan et al.'s experiment where all the NEs annotated by the BBN corpus were collapsed, Rajkumar et al. chose to collapse into single tokens only NEs whose exact form can be reasonably expected to be specified in the input to the realizer. For example, while some quantificational or comparatives phrases like *more than $ 10,000* are annotated as MONEY in the BBN corpus, Rajkumar et al. only collapse *$_10,000* into an atomic unit, with *more than* handled compositionally according to the semantics assigned to it by the grammar. Thus, after transferring the BBN annotations to the CCGbank corpus, Rajkumar et al. (partially) collapsed NEs which are CCGbank constituents according to the following rules: (1) completely collapse the PERSON, ORGANIZATION, GPE, WORK OF ART major class type entitites; (2) ignore phrases like *three decades later*, which are annotated as DATE entities; and (3) collapse all phrases with POS tags CD or NNP(S) or lexical items % or $, ensuring that all prototypical named entities are collapsed.

It is worth noting that improvements in our corpus-based grammar engineering process—including a more precise treatment of punctuation, better named entity handling and the addition of catch-all logical form templates—have resulted in a 13.5 BLEU point improvement in our baseline realization scores on Section 00 of the CCGbank, from a score of 0.6567 in (Espinosa et al., 2008) to 0.7917 in (Rajkumar et al., 2009), contributing greatly to the state-of-the-art results reported

in Section 4. A further 4.5 point improvement is obtained from the use of named entity classes in language modeling and hypertagging (Rajkumar et al., 2009), as described next, and from our perceptron reranking model, described in Section 3.

2.3 Factored Language Models

As in (White et al., 2007; Rajkumar et al., 2009), we use factored language models (Bilmes and Kirchhoff, 2003) over words, part-of-speech tags and supertags[1] to score partial and complete realizations. The trigram models were created using the SRILM toolkit (Stolcke, 2002) on the standard training sections (02–21) of the CCGbank, with sentence-initial words (other than proper names) uncapitalized. While these models are considerably smaller than the ones used in (Langkilde-Geary, 2002; Velldal and Oepen, 2005), the training data does have the advantage of being in the same domain and genre. The models employ interpolated Kneser-Ney smoothing with the default frequency cutoffs. The best performing model interpolates three component models using rank-order centroid weights: (1) a word trigram model; (2) a word model with semantic classes replacing named entities; and (3) a trigram model that chains a POS model with a supertag model, where the POS model (P) conditions on the previous two POS tags, and the supertag model (S) conditions on the previous two POS tags as well as the current one, as shown below:

$$p^{PS}(\vec{F}_i \mid \vec{F}_{i-2}^{i-1}) = p(P_i \mid P_{i-2}^{i-1})p(S_i \mid P_{i-2}^{i}) \quad (1)$$

Training data for the semantic class–replaced model was created by replacing (collapsed) words with their NE classes, in order to address data sparsity issues caused by rare words in the same semantic class. For example, the Section 00 sentence *Pierre_Vinken , 61 years old , will join the board as a nonexecutive director Nov._29 .* becomes *PERSON , DATE:AGE DATE:AGE old , will join the ORG_DESC:OTHER as a nonexecutive PER_DESC DATE:DATE DATE:DATE .* During realization, word forms are generated, but are then replaced by their semantic classes for scoring using the semantic class–replaced model, similar to Oh and Rudnicky (2002).

Note that the use of supertags in the factored language model to score possible realizations is distinct from the prediction of supertags for lexical category assignment: the former takes the words in the local context into account (as in supertagging for parsing), while the latter takes features of the logical form into account. This latter process we call hypertagging, to which we now turn.

2.4 Hypertagging

A crucial component of the OpenCCG realizer is the *hypertagger* (Espinosa et al., 2008), or supertagger for surface realization, which uses a maximum entropy model to assign the most likely lexical categories to the predicates in the input logical form, thereby greatly constraining the realizer's search space.[2] Figure 2 shows gold-standard supertags for the lexical predicates in the graph; such category labels are predicted by the hypertagger at run-time. As in recent work on using supertagging in parsing, the hypertagger operates in a multitagging paradigm (Curran et al., 2006), where a variable number of predictions are made per input predicate. Instead of basing category assignment on linear word and POS context, however, the hypertagger predicts lexical categories based on contexts within a directed graph structure representing the logical form (LF) of the sentence to be realized. The hypertagger generalizes Bangalore and Rambow's (2000) method of using supertags in generation by using maximum entropy models with a larger local context.

During realization, the hypertagger returns a β-best list of supertags in order of decreasing probability. Increasing the number of categories returned clearly increases the likelihood that the most-correct supertag is among them, but at a corresponding cost in chart size. Accordingly, the hypertagger begins with a highly restrictive value for β, and backs off to progressively less-restrictive values if no complete realization can be found using the set of supertags returned. Clark and Curran (2007b) have shown this iterative relaxation strategy to be highly effective in CCG parsing.

3 Perceptron Reranking

As Collins (2002) observes, perceptron training involves a simple, on-line algorithm, with few iterations typically required to achieve good performance. Moreover, *averaged* perceptrons—which

[1] With CCG, supertags (Bangalore and Joshi, 1999) are lexical categories considered as fine-grained syntactic labels.

[2] The approach has been dubbed *hypertagging* since it operates at a level "above" the syntax, moving from semantic representations to syntactic categories.

Input: training examples (x_i, y_i)
Initialization: set $\overline{\alpha} = 0$, or use optional input model
Algorithm:
for $t = 1 \ldots T, i = 1 \ldots N$
$z_i = \text{argmax}_{y \in \text{GEN}(x_i)} \Phi(x_i, y) \cdot \overline{\alpha}$
if $z_i \neq y_i$
$\overline{\alpha} = \overline{\alpha} + \Phi(x_i, y_i) - \Phi(x_i, z_i)$
Output: $\overline{\alpha} = \sum_{t=1}^{T} \sum_{i=1}^{N} \overline{\alpha}_{ti} / TN$

Figure 3: Averaged perceptron training algorithm

approximate voted perceptrons, a maximum-margin method with attractive theoretical properties—seem to work remarkably well in practice, while adding little further complexity. Additionally, since features only take on non-zero values when they appear in training items requiring updates, perceptrons integrate feature selection with, and often produce quite small models, especially when starting with a good baseline.

The generic averaged perceptron training algorithm appears in Figure 3. In our case, the algorithm trains a model for reranking the n-best realizations generated using our existing factored language model for scoring, with the oracle-best realization considered the correct answer. Accordingly, the input to the algorithm is a list of pairs (x_i, y_i), where x_i is a logical form, GEN(x_i) are the n-best realizations for x_i, and y_i is the oracle-best member of GEN(x_i). The oracle-best realization is determined using a 4-gram precision metric (approximating BLEU) against the reference sentence.

We have followed Huang (2008) in using oracle-best targets for training, rather than gold standard ones, in order to better approximate test conditions during training. However, following Clark & Curran (2007a), during training we seed the realizer with the gold-standard supertags, augmenting the hypertagger's β-best list, in order to ensure that the n-best realizations are generally of high quality; consequently, the gold standard realization (i.e., the corpus sentence) usually appears in the n-best list.[3] In addition, we use a hypertagger trained on all the training data, to improve hypertagger performance, while excluding the current training section (in jack-knifed fashion) from the word-based parts of the language model, in order to make the language model scores more realistic. It remains for future work to determine whether using a different compromise between ensuring high-quality training data and remaining faithful to the test conditions would yield better results.

Since realization of the n-best lists for training is the most time-consuming part of the process, in our current implementation we perform this step once, generating event files along the way containing feature vectors for each candidate realization. The event files are used to calculate the frequency distribution for the features, and minimum cutoffs are chosen to trim the feature alphabet to a reasonable size. Training then takes place by iterating over the event files, ignoring features that do not appear in the alphabet. As Figure 3 indicates, training consists of calculating the top-ranked realization according to the current model $\overline{\alpha}$, and performing an update when the top-ranked realization does not match the oracle-best realization. Updates to the model add the feature vector $\Phi(x_i, y_i)$ for the missed oracle-best realization, and subtract the feature vector $\Phi(x_i, z_i)$ for the mistakenly top-ranked realization. The final model averages the models across the T iterations over the training data, and N test cases within each iteration.

Note that while training the perceptron model involves n-best reranking, realization with the resulting model can be viewed as forest rescoring, since scoring of all partial realizations is integrated into the realizer's beam search. In future work, we intend to investigate saving the realizer's packed charts, rather than event files, and integrating the unpacking of the charts with the perceptron training algorithm.

The features we employ in our perceptron models are of three kinds. First, as in the log-linear models of Velldal & Oepen and Nakanishi et al., we incorporate the log probability of the candidate realization's word sequence according to our factored language model as a single feature in the perceptron model. Since our language model linearly interpolates three component models, we also include the log prob from each component language model as a feature, so that the combination of these components can be optimized.

Second, we include syntactic features in our

[3] As in Clark & Curran's approach, we use a single β value during training, rather than iteratively loosening the β value; the chosen β value determines the size of the discrimation space.

Feature Type	Example
LexCat + Word	$\mathsf{s/s/np}$ + before
LexCat + POS	$\mathsf{s/s/np}$ + IN
Rule	$\mathsf{s}_{dcl} \rightarrow \mathsf{np}\ \mathsf{s}_{dcl}\backslash\mathsf{np}$
Rule + Word	$\mathsf{s}_{dcl} \rightarrow \mathsf{np}\ \mathsf{s}_{dcl}\backslash\mathsf{np}$ + bought
Rule + POS	$\mathsf{s}_{dcl} \rightarrow \mathsf{np}\ \mathsf{s}_{dcl}\backslash\mathsf{np}$ + VBD
Word-Word	$\langle$company, $\mathsf{s}_{dcl} \rightarrow \mathsf{np}\ \mathsf{s}_{dcl}\backslash\mathsf{np}$, bought$\rangle$
Word-POS	$\langle$company, $\mathsf{s}_{dcl} \rightarrow \mathsf{np}\ \mathsf{s}_{dcl}\backslash\mathsf{np}$, VBD$\rangle$
POS-Word	$\langle$NN, $\mathsf{s}_{dcl} \rightarrow \mathsf{np}\ \mathsf{s}_{dcl}\backslash\mathsf{np}$, bought$\rangle$
Word + Δ_w	$\langle$bought, $\mathsf{s}_{dcl} \rightarrow \mathsf{np}\ \mathsf{s}_{dcl}\backslash\mathsf{np}\rangle$ + d_w
POS + Δ_w	$\langle$VBD, $\mathsf{s}_{dcl} \rightarrow \mathsf{np}\ \mathsf{s}_{dcl}\backslash\mathsf{np}\rangle$ + d_w
Word + Δ_p	$\langle$bought, $\mathsf{s}_{dcl} \rightarrow \mathsf{np}\ \mathsf{s}_{dcl}\backslash\mathsf{np}\rangle$ + d_p
POS + Δ_p	$\langle$VBD, $\mathsf{s}_{dcl} \rightarrow \mathsf{np}\ \mathsf{s}_{dcl}\backslash\mathsf{np}\rangle$ + d_p
Word + Δ_v	$\langle$bought, $\mathsf{s}_{dcl} \rightarrow \mathsf{np}\ \mathsf{s}_{dcl}\backslash\mathsf{np}\rangle$ + d_v
POS + Δ_v	$\langle$VBD, $\mathsf{s}_{dcl} \rightarrow \mathsf{np}\ \mathsf{s}_{dcl}\backslash\mathsf{np}\rangle$ + d_v

Table 1: Basic and dependency features from Clark & Curran's (2007b) normal form model; distances are in intervening words, punctuation marks and verbs, and are capped at 3, 3 and 2, respectively

model by implementing Clark & Curran's (2007b) normal form model in OpenCCG.[4] The features of this model are listed in Table 1; they are integer-valued, representing counts of occurrences in a derivation. These syntactic features are quite comparable to the dominance-oriented features in the union of the Velldal & Oepen and Nakanishi et al. models, except that our feature set does not include grandparenting, which has been found to have limited utility in CCG parsing. Our syntactic features also include ones that measure the distance between headwords in terms of intervening words, punctuation marks or verbs; these features generalize the ones in Nakanishi et al.'s model. Note that in contrast to parsing, in realization distance features are non-local, since different partial realizations in the same equivalence class typically differ in word order; as we are working in a reranking paradigm though, the non-local nature of these features is unproblematic.

Third, we include discriminative n-gram features in our model, following Roark et al.'s (2004) approach to discriminative n-gram modeling for speech recognition. By discriminative n-gram features, we mean features counting the occurrences of each n-gram that is scored by our factored language model, rather than a feature whose value is the log prob determined by the language model. As Roark et al. note, discriminative training with n-gram features has the potential to learn to negatively weight n-grams that appear in some of the GEN(x_i) candidates, but which never appear in the naturally occurring corpus used to train a standard, generative language model. Since our factored language model considers words, semantic classes, part-of-speech tags and supertags, our n-gram features represent a considerable generalization of the sequence-oriented features in Velldal & Oepen's model, which never contain more than one word and do not include semantic classes.

[4] We have omitted Clark & Curran's root features, since the category we use for the full stop ensures that it must appear at the root of any complete derivation.

Model	#Alph-feats	#Feats	Acc	Time
full-model	2402173	576176	96.40%	08:53
lp-ngram	1127437	342025	94.52%	05:19
lp-syn	1274740	291728	85.03%	05:57

Table 2: Perceptron Training Details—number of features in the alphabet, number of features in the model, training accuracy and training time (hours) for 10 iterations on a single commodity server

4 Evaluation

4.1 Experimental Conditions

For the experiments reported below, we used a lexico-grammar extracted from Sections 02–21 of our enhanced CCGbank, a hypertagging model incorporating named entity class features, and a trigram factored language model over words, named entity classes, part-of-speech tags and supertags, as described in the preceding section. BLEU scores were calculated after removing the underscores between collapsed NEs.

Events were generated for each training section separately. As already noted, the hypertagger and POS/supertag language model was trained on all the training sections, while separate word-based models were trained excluding each of the training sections in turn. Event files for 26530 training sentences with complete realizations were generated in 7 hours and 16 minutes on a cluster using one commodity server per section, with an average n-best list size of 18.2. Perceptron models were trained on single machines; details for three of the models appear in Table 2. The complete set of models is listed in Table 3.

4.2 Results

Realization results on the development section are given in Table 4. As the first block of rows after the baseline shows, of the models incorporating a single kind of feature, only the one with the n-gram log prob features beats the baseline BLEU

Model	Description
baseline-w3	No perceptron (3g wd only)
baseline	No perceptron
syn-only-nodist	All syntactic features except distance
ngram-only	Just ngram features
syn-only	Just syntactic features
lp-only	Just log prob features
lp-ngram	Log prob + Ngram features
lp-syn	Log prob + Syntactic features
full-model	Log prob + Ngram +Syntactic features

Table 3: Legend for Experimental Conditions

score, with the other models falling well below the baseline (though faring better than the trigram-word LM baseline). This result confirms the importance of including n-gram log prob features in discriminative realization ranking models, in line with Velldal & Oepen's findings, and contra those of Nakanishi et al., even though it was Nakanishi et al. who experimented with the Penn Treebank corpus, while Velldal & Oepen's experiments were on a much smaller, limited domain corpus. The second block of rows shows that both the discriminative n-gram features and the syntactic features provide a substantial boost when used with the n-gram log prob features, with the syntactic features yielding a more than 3 BLEU point gain. The final row shows that the full model works best, though the boosts provided by the syntactic and discriminative n-gram features are clearly not independent. The BLEU point trends are mirrored in the percentage of exact match realizations, which goes up by more than 10% from the baseline. The percentage of complete (i.e., non-fragmentary) realizations, however, goes down; we expect that this is due to the time taken up by our current naive method of feature extraction, which does not cache the features calculated for partial realizations. Realization results on the standard test section appear in Table 5, confirming the gains made by the full model over the baseline.[5]

We calculated statistical significance for the main results on the development section using bootstrap random sampling.[6] After re-sampling 1000 times, significance was calculated using a paired t-test (999 d.f.). The results indicated that lp-only exceeded the baseline, lp-ngram and lp-syn exceeded lp-only, and the full model exceeded lp-syn, with $p < 0.0001$ in each case.

[5]Note that the baseline for Section 23 uses 4-grams and a filter for balanced punctuation (White and Rajkumar, 2008), unlike the other reported configurations, which would explain the somewhat smaller increase seen with this section.

[6]Scripts for running these tests are available at `http://projectile.sv.cmu.edu/research/public/tools/bootStrap/tutorial.htm`

Model	%Exact	%Compl.	BLEU	Time
baseline-w3	26.00	83.15	0.7646	1.8
baseline	29.00	83.28	0.7963	2.0
syn-only-nodist	26.02	82.69	0.7754	3.2
ngram-only	27.67	82.95	0.7777	3.0
syn-only	28.34	82.74	0.7838	3.4
lp-only	32.01	83.02	0.8009	2.1
lp-ngram	36.31	80.47	0.8183	3.1
lp-syn	39.47	79.74	0.8323	3.5
full-model	**40.11**	79.63	**0.8373**	3.6

Table 4: Section 00 Results (98.9% coverage)—percentage of exact match and grammatically complete realizations, BLEU scores and average times, in seconds

Model	%Exact	%Complete	BLEU
baseline	33.74	85.04	0.8173
full-model	**40.45**	83.88	**0.8506**

Table 5: Section 23 Results (97.06% coverage)

4.3 Examples

Table 6 presents four examples where the full model improves upon the baseline. Example sentence wsj_0020.10 in Table 6 is a case where the perceptron successfully weights the component ngram models, as the lp-ngram model and those that build on it get it right. Note that here, the modifier ordering in *small video-viewing* is not specified in the logical form and either ordering is possible syntactically. In wsj_0024.2, number agreement between the conjoined subject noun phrase and verb is obtained only with the full model. This suggests that the full model is more robust to cases where the grammar is insufficiently precise (number agreement is enforced by the grammar in only the simplest cases). Example wsj_0034.9 corrects a VP ordering mismatch, where the corpus sentence is clearly preferred to the one where *into oblivion* is shifted to the end. Finally, wsj_0047.13 corrects an animacy mismatch on the *wh*-pronoun, in large part due to the high negative weight assigned to the discriminative n-gram feature PERSON_,_which. Note that the full model still differs from the original sentence in its placement of the adverb *reportedly*, choosing the arguably more natural position following the auxiliary.

4.4 Comparison to Other Systems

Table 7 lists our results in the context of those reported for other systems on PTB Section 23. The

Ref-wsj_0020.10	that measure could compel Taipei 's growing number of small video-viewing parlors to pay ...
baseline,syn-only,ngram-only	that measure could compel Taipei 's growing number of *video-viewing small* parlors to ...
lp-only, lp-ngram, full-model	that measure could compel Taipei 's growing number of **small video-viewing** parlors to ...
Ref-wsj_0024.2	Esso Australia Ltd. , a unit of new york-based Exxon Corp. , and Broken Hill Pty. operate the fields ...
all except full-model	Esso Australia Ltd. , a unit of new york-based Exxon Corp. , and Broken Hill Pty. *operates* the fields ...
full-model	Esso Australia Ltd. , a unit of new york-based Exxon Corp. , and Broken Hill Pty. **operate** the fields ...
Ref-wsj_0034.9	they fell into oblivion after the 1929 crash .
baseline, lp-ngram	they fell *after the 1929 crash into oblivion* .
lp-only, ngram-only, syn-only, full-model	they fell **into oblivion after the 1929 crash** .
Ref-wsj_0047.13	Antonio Novello , whom Mr. Bush nominated to serve as surgeon general , reportedly has assured ...
baseline,baseline-w3, lp-syn, lp-only	Antonio Novello , *which* Mr. Bush nominated to serve as surgeon general , has *reportedly* assured ...
full-model, lp-ngram, syn-only, ngram-syn	Antonio Novello , **whom** Mr. Bush nominated to serve as surgeon general , has *reportedly* assured ...

Table 6: Examples of realized output

System	Coverage	BLEU	%Exact
Callaway (05)	98.5%	0.9321	57.5
OpenCCG (09)	97.1%	0.8506	40.5
Ringger et al. (04)	100.0%	0.836	35.7
Langkilde-Geary (02)	83%	0.757	28.2
Guo et al. (08)	100.0%	0.7440	19.8
Hogan et al. (07)	≈100.0%	0.6882	-
OpenCCG (08)	96.0%	0.6701	16.0
Nakanishi et al. (05)	90.8%	0.7733	-

Table 7: PTB Section 23 BLEU scores and exact match percentages in the NLG literature (Nakanishi et al.'s results are for sentences of length 20 or less)

most similar systems to ours are those of Nakanishi et al. (2005) and Hogan et al. (2007), as they both involve chart realizers for reversible grammars engineered from the Penn Treebank. While direct comparisons across systems cannot really be made when inputs vary in their semantic depth and specificity, we observe that our all-sentences BLEU score of 0.8506 exceeds that of Hogan et al., who report a top score of 0.6882 (though with coverage near 100%), and also surpasses Nakanishi et al.'s score of 0.7733, despite their results being limited to sentences of length 20 or less (with 91% coverage). Velldal & Oepen's (2005) system is also closely related, as noted in the introduction, but as their experiments are on a limited domain corpus, their results cannot be compared at all meaningfully.

5 Related Work and Discussion

As alluded to above, realization systems cannot be easily compared, even on the same corpus, when their inputs are not the same. This point is dramatically illustrated in Langkilde-Geary's (2002) system, where a BLEU score of 0.514 is reported for minimally specified inputs on PTB Section 23, while a score of 0.757 is reported for the 'Permute, no dir' case (which perhaps most closely resembles our inputs), and a score of 0.924 is reported for the most fully specified inputs; note, however, that in the latter case word order is determined by sibling order in the inputs, an assumption not commonly made. As another example, Guo et al. (2008) report a competitive result of 0.7440 (with 100% coverage) using a dependency-based approach; however, their inputs, like those of Hogan et al., include more surface syntactic information than ours, as they specify case-marking prepositions, *wh*-pronouns and complementizers. In a recent experiment to assess the impact of input specificity, we found that including predicates for all prepositions in our logical forms boosted our baseline results by more than 3 BLEU points, with complete realizations found in more than 90% of the test cases, indicating that generating from a more surfacy input is indeed an easier task than generating from a deeper representation. Given the current lack of consensus on realizer input specificity, we believe it is important to keep in mind that within-system comparisons (such as those in the preceding section) are the ones that should be given the most credence.

Returning to our cross-system comparison, it is perhaps surprising that Callaway (2005) reports the best PTB BLEU score to date, 0.9321, with 98.5% coverage, using a purely symbolic, handcrafted grammar augmented to handle the most frequent coverage issues for the PTB. While Callaway's inputs are unordered, word order is often determined by positional features (e.g. `front`) or by the type of modification (e.g. `describer` vs. `qualifier`), and parts-of-speech are included for lexical items. Additionally, in contrast to our approach, Callaway makes use of a generation-only grammar, rather than a reversible one, and his approach is less well-suited to producing n-best

outputs. Nevertheless, his high scores do suggest the potential for precise grammar engineering to improve realization quality.

While we have yet to perform a thorough error analysis, our impression is that although the current set of syntactic features substantially improves clausal constituent ordering, a variety of disfluent cases remain. More thorough investigations of features for constituent ordering in English have been performed by Ringger et al. (2004), Filippova and Strube (2009) and Zhong and Stent (2009), all of whom develop classifiers for determining linear order. In future work, we plan to investigate whether features inspired by these approaches can be usefully integrated into our perceptron reranker.

Also related to the present work is discriminative training in syntax-based MT (Turian et al., 2007; Watanabe et al., 2007; Blunsom et al., 2008; Chiang et al., 2009). Not surprisingly, since MT is a harder problem than surface realization, syntax-based MT systems have made use of less precise grammars and more impoverished (target-side) feature sets than those tackling realization ranking. With progress on discriminative training with large numbers of features in syntax-based MT, the features found to be useful for high-quality surface realization may become increasingly relevant for MT as well.

6 Conclusions

In this paper, we have shown how discriminative reranking with an averaged perceptron model can be used to achieve substantial improvements in realization quality with CCG. Using a comprehensive feature set, we have also confirmed the utility of including language model log probabilities as features in the model, which prior work on discriminative training with log linear models for HPSG realization had called into question. The perceptron model allows the combination of multiple n-gram models to be optimized and then augmented with both syntactic features and discriminative n-gram features, inspired by related work in discriminative parsing and language modeling for speech recognition. The full model yields a state-of-the-art BLEU score of 0.8506 on Section 23 of the CCGbank, to our knowledge the best score reported to date using a reversible, corpus-engineered grammar, despite our use of deeper, less specific inputs. Finally, the perceptron model paves the way for exploring the utility of richer feature spaces in statistical realization, including the use of linguistically-motivated and non-local features, a topic which we plan to investigate in future work.

Acknowledgements

This work was supported in part by NSF grant IIS-0812297 and by an allocation of computing time from the Ohio Supercomputer Center. Our thanks also to the OSU Clippers group and the anonymous reviewers for helpful comments and discussion.

References

Jason Baldridge and Geert-Jan Kruijff. 2002. Coupling CCG and Hybrid Logic Dependency Semantics. In *Proc. ACL-02*.

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational Linguistics*, 25(2):237–265.

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proc. COLING-00*.

Jeff Bilmes and Katrin Kirchhoff. 2003. Factored language models and general parallelized backoff. In *Proc. HLT-03*.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proc. ACL-08: HLT*.

Johan Bos. 2005. Towards wide-coverage semantic interpretation. In *Proc. IWCS-6*.

Stephen Boxwell and Michael White. 2008. Projecting Propbank roles onto the CCGbank. In *Proc. LREC-08*.

Charles Callaway. 2005. The types and distributions of errors in a wide coverage surface realizer evaluation. In *Proceedings of the 10th European Workshop on Natural Language Generation*.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proc. NAACL HLT 2009*.

Stephen Clark and James Curran. 2007a. Perceptron training for a wide-coverage lexicalized-grammar parser. In *ACL 2007 Workshop on Deep Linguistic Processing*.

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

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. ACL-04*.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proc. EMNLP-02*.

James R. Curran, Stephen Clark, and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proc. COLING/ACL-06*.

Dominic Espinosa, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proc. ACL-08: HLT*.

Katja Filippova and Michael Strube. 2009. Tree linearization in English: Improving language model based approaches. In *Proc. NAACL HLT 2009 Short Papers*.

Yuqing Guo, Josef van Genabith, and Haifeng Wang. 2008. Dependency-based n-gram models for general purpose sentence realisation. In *Proc. COLING-08*.

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

Deirdre Hogan, Conor Cafferkey, Aoife Cahill, and Josef van Genabith. 2007. Exploiting multi-word units in history-based probabilistic generation. In *Proc. EMNLP-CoNLL*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. ACL-08: HLT*.

Irene Langkilde-Geary. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proc. INLG-02*.

Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic methods for disambiguation of an HPSG-based chart generator. In *Proc. IWPT-05*.

Alice H. Oh and Alexander I. Rudnicky. 2002. Stochastic natural language generation for spoken dialog systems. *Computer, Speech & Language*, 16(3/4):387–407.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: A corpus annotated with semantic roles. *Computational Linguistics*, 31(1).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, PA.

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

Rajakrishnan Rajkumar, Michael White, and Dominic Espinosa. 2009. Exploiting named entity classes in CCG surface realization. In *Proc. NAACL HLT 2009 Short Papers*.

Eric Ringger, Michael Gamon, Robert C. Moore, David Rojas, Martine Smets, and Simon Corston-Oliver. 2004. Linguistically informed statistical models of constituent structure for ordering in sentence realization. In *Proc. COLING-04*.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proc. ACL-04*.

Mark Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA, USA.

Andreas Stolcke. 2002. SRILM — An extensible language modeling toolkit. In *Proc. ICSLP-02*.

Joseph Turian, Benjamin Wellington, and I. Dan Melamed. 2007. Scalable discriminative learning for natural language parsing and translation. In *Proc. NIPS 19*.

Erik Velldal and Stephan Oepen. 2005. Maximum entropy models for realization ranking. In *Proc. MT Summit X*.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proc. EMNLP-CoNLL-07*.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Technical report, BBN.

Michael White and Rajakrishnan Rajkumar. 2008. A more precise analysis of punctuation for broad-coverage surface realization with CCG. In *Proc. of the Workshop on Grammar Engineering Across Frameworks (GEAF08)*.

Michael White, Rajakrishnan Rajkumar, and Scott Martin. 2007. Towards broad coverage surface realization with CCG. In *Proc. of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+MT)*.

Michael White. 2006. Efficient Realization of Coordinate Structures in Combinatory Categorial Grammar. *Research on Language and Computation*, 4(1):39–75.

Huayan Zhong and Amanda Stent. 2009. Determining the position of adverbial phrases in English. In *Proc. NAACL HLT 2009 Short Papers*.

Multi-Document Summarisation Using Generic Relation Extraction

Ben Hachey
Centre for Languate Tecnology
Macquarie University
NSW 2109 Australia

Capital Markets CRC Limited
GPO Box 970
Sydney NSW 2001

bhachey@cmcrc.com

Abstract

Experiments are reported that investigate the effect of various source document representations on the accuracy of the sentence extraction phase of a multi-document summarisation task. A novel representation is introduced based on generic relation extraction (GRE), which aims to build systems for relation identification and characterisation that can be transferred across domains and tasks without modification of model parameters. Results demonstrate performance that is significantly higher than a non-trivial baseline that uses *tf*idf*-weighted words and at least as good as a comparable but less general approach from the literature. Analysis shows that the representations compared are complementary, suggesting that extraction performance could be further improved through system combination.

1 Introduction

The goal of summarisation is to take an information source, extract content from it, and present the most important content in a condensed form (Mani, 2001). The field of automatic summarisation (Mani, 2001; Spärck Jones, 2007) aims to create tools that address various summarisation tasks with minimal human intervention. Extractive approaches to automatic summarisation create representations of the source document that are generally based on an easily identified text sub-unit such as sentences or paragraphs. These representations are then used to identify representative or otherwise important snippets of text to place in the summary.

Following Spärck Jones (2007), summarisation systems can be characterised with respect to their approach to three main sub-tasks: 1) interpretation, 2) transformation and 3) generation. The input consists of the source document (or a collection of source documents in the case of multi-document summarisation). The first step (interpretation) creates a representation of the source document by performing some level of interpretation. A simple approach here represents sentences by their tokens (i.e., as an unordered bag-of-words). The next step (transformation) is the compaction step where the source representation is converted into the summary representation, e.g. by identifying sentences whose words are most representative of the full text. Finally, in the last step (generation), the output summary is created. In the case of sentence extraction, this includes various operations to maximise coherence such as ensuring that entity references are comprehensible and arranging the sentences in a sensible order.

The current work investigates several representations of source documents. In particular, an approach from the literature based on atomic events (Filatova and Hatzivassiloglou, 2004) is compared to a novel approach based on generic relation extraction (GRE), which aims to build systems for relation identification and characterisation that can be transferred across domains and tasks without modification of model parameters (Hachey, 2009). The various representations are substituted in the interpretation phase of a multi-document summarisation task and used as the basis for extracting sentences to be placed in the summary. System summaries are compared by calculating term overlap with reference summaries created by human analysts.

2 Motivation

In seminal work on automatic summarisation, Luhn (1958) introduces a representation based on content words. These are defined as non-function words from the source document that are neither too frequent nor too infrequent. Luhn uses frequency to weight content words and ex-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 420–429,
Singapore, 6-7 August 2009.

tracts sentences with the highest combined content scores to form the summary. Subsequent work adapted the *tf*idf* weighting scheme, where term frequency (*tf*) is combined with inverse document frequency (*idf*), an inverse measure of term occurrence across documents that serves to downweight common words (Spärck Jones, 1972). In modern work, *tf*idf* representations are often used as simple but non-trivial baselines. The problem is that these shallow features often break down where underlying linguistic content needs to be compared rather than just surface structure.

The use of representations based on information extraction (IE) has been suggested as one approach to capturing deeper semantic information. This is based on the notion that IE definitions of types for entities, relations and events provide a level of abstraction that is appropriate for automatic summarisation. Several approaches in the literature have explored the use of IE-based representations for extractive summarisation: McKeown et al. (1998) incorporate patient characteristic templates for matching potential treatments to specific patients in a medical summarisation system; White and Cardie (2002) incorporate a bootstrapped IE system based on Autoslog (Riloff, 1996) for filling event templates; and Harabagiu and Maiorano (2002) incorporate a hybrid approach that uses conventional supervised IE techniques for known topics and a more general approach based on WordNet for unknown topics.[1] The problem with these systems is that they all use supervised approaches to IE that require that the IE templates be known in advance and additionally require significant investment in writing extraction rules or in annotating data for training. Where more general techniques are used, they still require domain-specific resources, e.g. White and Cardie (2002) bootstrapping approach still requires that the extraction templates be known in advance and Harabagiu and Maiorano (2002) approach depends on the WordNet lexical database, for which coverage is not guaranteed for arbitrary domains.

Filatova and Hatzivassiloglou (2004) introduce methods using more general IE representations that are not based on supervised learning. Given a named entity recogniser, the representation is automatically derived and consists of $<Ent, Connector, Ent>$ event triples, where connectors are verbs or action nouns that occur in between the two NEs. Thus, the approach aims to perform a generic IE task that the authors refer to as atomic event extraction. This representation is shown to outperform a *tf*idf* baseline on a multi-document summarisation task. As we will see in Section 4.3 below, Filatova and Hatzivassiloglou's approach has three main shortcomings. First, it focuses exclusively on simple atomic events (i.e., entity mention pairs with an intervening verbal connector), meaning that it will not be able to address tasks where relations are at least as important as events (e.g., biographical summarisation). Second, it relies on exact matching between connectors, which is not capable of capturing latent semantic similarities (e.g., between 'work for' and 'employed by'). Third, its performance is subject to the coverage of WordNet, which is used to identify action nouns.

Generic relation extraction (GRE) aims to build systems that can be transferred across domains and tasks without modification of model parameters (Hachey, 2009). For relation identification (i.e., extraction of relation forming entity mention pairs), this is achieved by using general rule-based approaches and, for relation characterisation (i.e., assignment of types to relation mentions), this is achieved by using unsupervised machine learning. Hachey (2009) introduces a GRE approach that addresses the shortcomings of the atomic event approach mentioned above. First, it models a type of IE that includes relations. Second, it uses a connector model based on latent Dirichlet allocation (Blei et al., 2003), which provides a mechanism for capturing latent semantic similarities between connectors. Third, it does not rely on domain-specific resource like WordNet. The GRE models used here do rely on dependency parsing. However, they still generalise across formal domains as the relation identification and characterisation systems, developed on news data, achieve comparable performance when applied directly to a relation extraction task in the biomedical domain (see Hachey (2009) for details). Furthermore, grammatical relations obtained from dependency parsing provide a means for constraining relation identification and supplying more linguistically meaningful features for relation characterisation.

[1] Comparable approaches using IE in the context of abstractive–as opposed to extractive–summarisation include work by DeJong (1982), Hahn and Reimer (1999), White et al. (2001) and Saggion and Lapalme (2002).

	c_1	c_2	c_3	c_4	c_5
t_1	1	1	0	1	1
t_2	1	0	0	1	0
t_3	0	1	0	0	1
t_4	1	0	1	1	1

Table 1: Text × concept matrix for set cover approach to automatic summarisation (Filatova and Hatzivassiloglou, 2004).

3 Algorithm for Set Cover Extraction

For the sake of comparison, the current evaluation adopts the Filatova and Hatzivassiloglou (2004) summarisation framework. This defines an extraction approach based on a mapping between *textual* units and *concepts*. To illustrate, consider the matrix in Table 1 where rows represent textual units (e.g., sentences, paragraphs) and columns represent concepts (e.g., words, events, relations) in the input text. Each concept is either absent or present in a given textual unit. Additionally, each concept has a weight associated with it. Looking at the problem in this way makes it natural to formulate it as follows: *the summary should select textual units such that there is maximal coverage of the salient conceptual units*.[2] This is essentially the maximum coverage problem, which has been shown to be reducible to the set covering problem, for which there are approximation algorithms in the literature that run in polynomial time or better (Hochbaum, 1997; Bienstock and Iyengar, 2004).

Filatova and Hatzivassiloglou define several greedy algorithms that can be parametrised in terms of the general SUMMARISE function in Figure 1, which takes the text × concept matrix D and the maximum summary length k as input. The SUMMARISE function first initialises the summary $\mathcal{S}$ to the empty set. Then it enters a loop that continues until the summary reaches the desired length. Within the loop, a text unit is extracted and added to the summary after which the text × concept matrix is updated The output of the algorithm is a set $\mathcal{S}$ comprising the text units that make up the summary. For the experiments reported here, the text units t are sentences and LENGTH(t_i) returns the count of word tokens in sentence t_i.

```
SUMMARISE : D, k
1  S ← {}
2  while Σ_{t_i∈S} LENGTH(t_i) < k
3        t_j ← EXTRACT(D)
4        S ← S ∪ t_j
5        D ← UPDATE(D, t_j)
6  return S
```

Figure 1: Generalised function for Filatova and Hatzivassiloglou (2004) approach to extractive summarisation.

```
EXTRACT: D
1  c_j ← arg max_{c_j∈cols(D)} Σ_{t_i∈rows(D)} D[t_i, c_j]
2  t_k ← arg max_{t_k∈rows(D)&D[t_k,c_j]>0} SCORE(D, t_k)
3  return t_k
```

```
UPDATE: D, t_i
1  for each c_j ∈ cols(D)
2        if D[t_i, c_j] > 0
3              for each t_k ∈ rows(D)
4                    D[t_k, c_j] ← 0
5  D ← DELETE(D, t_i)
6  return D
```

Figure 2: Extraction and update functions for Filatova and Hatzivassiloglou (2004) modified adaptive algorithm.

Figure 2 contains the EXTRACT and UPDATE functions used here.[3] The EXTRACT function first identifies the concept c_j not yet covered in the summary that has the highest overall weight in the text × concept matrix D. Then it selects the text unit t_k with the highest score from among the text units that contain concept c_j. The SCORE function is the sum of concept weights for the given text unit, i.e.:

$$\text{SCORE} : D, t_i \mapsto \textbf{return} \sum_{c_j \in cols(D)} D[t_i, c_j] \quad (1)$$

The UPDATE function in Figure 2 aims to minimise redundancy in the summary by globally maximising the number of conceptual units covered in the output. In addition to removing the row representing the extracted text unit from the text × concept matrix D, it iterates through the remaining text units and assigns zero weights to all concepts that are covered by the extracted text unit.

[2] While not considered in the current experiments, a more discourse-oriented approach could be derived within the set cover framework by down-weighting conceptual units that occur e.g. in portions of the source documents that describe background information, where text segments containing background information could be identified using a sentence-level rhetorical status classifier like that developed by Teufel and Moens (2002).

[3] The EXTRACT and UPDATE functions in Figure 2 correspond to Filatova and Hatzivassiloglou (2004) modified adaptive algorithm and were found in preliminary experiments to be the better than the simple greedy and adaptive greedy algorithms (see Hachey (2009) for details).

Bush worked as an oil lease negotiator for Amoco in Denver and later started his own oil company, JNB.
*tf*idf* (*TF*) jnb:3.55, amoco:3.13, oil:3.05, negotiator:3.04, lease:2.58, denver:2.45, bush:2.44, worked:2.28, started:2.21, later:2.13, own:1.96, company:1.94, ...
event (*EV*) <PER_bush,worked,XFN_oil>:0.00023, <PER_bush,worked,ORG_amoco>:0.00011, <PER_bush,worked,LOC_denver>:0.00011, <XFN_oil,started,ORG_jnb>:0.00011, ...
relation (*RL*) <ORG_amoco,rd94,LOC_denver>:0.00039, <ORG_amoco,rd505,LOC_denver>:0.00039, <XFN_oil,rd92,ORG_jnb>:0.00002, <XFN_oil,rd712,ORG_jnb>:0.00002, ...
entity pair$_{ev}$ (*EE*) <PER_bush,XFN_oil>:0.00244, <PER_bush,LOC_denver>:0.00122, <PER_bush,ORG_jnb>:0.00044, <LOC_denver,XFN_oil>:0.00033, ...
entity pair$_{rl}$ (*ER*) <ORG_amoco,LOC_denver>:0.00311, <ORG_jnb,XFN_oil>:0.00155

Figure 3: Example sentence and various representations of sentence content.

4 Models

Figure 3 contains an example sentence and its representations corresponding to the various models of sentence content explored here.[4] These are described in detail in the rest of this section.

4.1 Baseline *tf*idf* Representation (TF)

The baseline model represents sentences as *tf*idf*-weighted bags-of-words (*TF*). Document frequencies for terms are derived from the same resource used by Filatova and Hatzivassiloglou (2004)–a frequency list compiled from a large sample of web pages. Term weighting is calculated using *tf*idf* as:

$$w(i,j) = \sqrt{(1 + \log(tf_{i,j})) * \log\left(\frac{N}{df_i}\right)} \quad (2)$$

where $tf_{i,j}$ is the number of times term i occurs in sentence j and df_i is the number of documents in which term i occurs. An example sentence and its *tf*idf* representation can be seen in Figure 3.

[4]The sentence was selected from document set d47 (from the data set described in Section 5.1 below), which contains articles about Neil Bush and his role in the collapse of Silverado Savings and Loan during the U.S. Savings and Loan crisis of the 1980s and 1990s.

4.2 Event Representation (EV)

We also compare to Filatova and Hatzivassiloglou (2004) atomic events (*EV*). This consists of $<Ent_i, Connector_j, Ent_k>$ event triples, where *connectors* are verbs or action nouns (i.e., nouns that are hyponyms of event or activity in WordNet) that occur in between the two entity mentions. Given a named entity recogniser and a lexical resource (WordNet), these are derived automatically from the text as follows. In the first step, all pairs of entity mentions that occur together in a sentence are identified. Next, the algorithm characterises the entity mention pairs using the connector words from the intervening context and discards pairs without an intervening connector word.

Event triple weighting is calculated by combining entity pair and connector weights as:

$$w_{ev}(i,j,k) = w_{ne}(i,k) * w_{cn}(j,i,k) \quad (3)$$

where $w_{ne}(i,k)$ is the weight of the entity pair $<i,k>$ consisting of entities i and k and $w_{cn}(j,i,k)$ is the weight of connector j in the context of entity pair $<i,j>$. $w_{ne}(i,k)$ is calculated as the normalised entity pair count, i.e.:

$$w_{ne}(i,k) = \frac{C_{ne}(<i,k>)}{C_{ne}(<*,*>)} \quad (4)$$

where $C_{ne}(<i,k>)$ is the count of mentions of entity pair $<i,k>$[5] and $C_{ne}(<*,*>)$ is the total count of entity mention pairs. And, $w_{cn}(j,i,k)$ is calculated as the normalised count of connector j in the context of the entity pair, i.e.:

$$w_{cn}(j,i,k) = \frac{C_{cn}^{<i,k>}(j)}{C_{cn}^{<i,k>}(*)} \quad (5)$$

where $C_{cn}^{<i,k>}(j)$ is the count of occurrences of connector j in the context of entity pair $<i,k>$ and $C_{cn}^{<i,k>}(*)$ is the total count of connectors in the context of entity pair $<i,k>$. An example sentence and its *event* representation can be seen in Figure 3. Event triples generated include <PER_bush,worked,ORG_amoco> and <PER_bush,started,ORG_jnb>.

Some erroneous event triples are also generated. The first error has to do with the fact that entities

[5]Coreference between entity mentions is computed by exact string match after removing punctuation, converting to all lower case, and prefixing the entity type. For example, the entity mention string "JNB" with type ORGANISATION is normalised to ORG_jnb.

include named entities identified in the preprocessing as well as the ten most frequent nouns in the document set. In the example sentence from Figure 3, the most frequent nouns include 'oil' but not 'negotiator' or 'company'. Therefore, 'oil' is labelled as an entity and extracted in a number of triples such as <PER_bush,worked, XFN_oil> (as opposed to <PER_bush, worked,XFN_negotiator>). Another problem illustrated by the example sentence has to do with the noisy nature of the surface-level approach to identifying entity mention pairs and connectors which tends to generate many false positive events, e.g. <ORG_amoco,started, ORG_jnb>. If the algorithm was constrained based on the underlying grammatical structure, it should be able to identity that the arguments of 'worked' are 'Bush' and 'Amoco' (i.e., <PER_bush,worked,ORG_amoco>) and that 'worked' does not describe an event involving 'Amoco' and 'JNB'.

4.3 Relation Representation (RL)

The focus of the current evaluation is a novel representation based on generic relation extraction (GRE). As mentioned above, GRE is a minimally supervised approach to the relation extraction task that aims to build systems for relation identification and characterisation that can be transferred across domains and tasks without modification of model parameters. Relation mentions are identified by taking pairs of entity mentions that have either 1) no more than two intervening words in the surface order of the sentence or 2) no more than one edge intervening on the shortest path through a dependency parse (see Hachey (2009) for details and experiments comparing different window configurations). This is stricter than the Filatova and Hatzivassiloglou approach in that entity mentions have to occur much closer or be connected by a single dependency relation. At the same time, it is less strict in the sense that an action- or event-denoting word is not required in the context, which makes it a more general model of IE.

Relation *connectors* are derived from a model of relation types based on latent Dirichlet allocation (Blei et al., 2003) that incorporates word, entity and dependency path features from the context of a relation-forming entity mention pair (see Hachey (2009) for details). This outputs a topic distribution for each entity mention pair that corresponds to the type of relation that is described. This representation 1) models a type of generic IE that includes relations, 2) uses a connector model that abstracts away from surface-level event descriptors used by Filatova and Hatzivassiloglou (2004) and 3) does not rely on domain-specific resources like WordNet.[6] For the purpose of comparison, relation triples are weighted in the same way as event triples using Equations 3 and 4 above. However, the connector pair weighting is modified to use the distribution over topics given by the LDA output.[7]

Relation triples generated for the example sentence in Figure 3 include <ORG_amoco, rd94,LOC_denver> and <ORG_amoco, rd505,LOC_denver>, where the connectors (i.e., rd94 and rd505) are identifiers that index particular topics from the LDA output. Here, rd94 and rd505 index topics that correspond to *located-in* relations so the respective triples both describe *located-in* relations between Amoco and Denver. Relation triples generated for the example sentence also include <XFN_oil, rd92,ORG_jnb> and <XFN_oil,rd712, ORG_jnb>. These are erroneous for the same reason as some of the event triples above (i.e., due to the noise inherent in the approach to identifying nominal entity mentions by identifying the ten most frequent nouns in the document set).

4.4 Entity Pair Representations (EE, ER)

Finally, we investigate the performance of representations that do not model event or relation type information. These are identical to the EV and RL representations above, except they are $<Ent, Ent>$ 2-tuples instead of $<Ent, Connector, Ent>$ 3-tuples. That is, entity pairs are included here provided that they meet the relation mention identification constraints. They are weighted using the normalised entity pair count (Equation 4 above). Relation-based entity pairs generated for the example sentence in Figure 3 include <LOC_denver,ORG_amoco> and <ORG_jnb,XFN_oil>.

[6] The GRE representation here does rely on dependency parsing, however, Hachey (2009) shows that it is still directly portable between the news and biomedical domains without modification of model parameters.

[7] Distributions for entity mention pairs tend to have a long uniform tail and only a few topics with higher probability. In converting to a weighting scheme, topic representations here are converted to a sparse representation where all topics in the uniform tail are removed.

5 Experimental Setup

5.1 Data

The experiments here use the multi-document summarisation data from the 2001 Document Understanding Conference (DUC),[8] which is the same data used by Filatova and Hatzivassiloglou (2004). This comprises 30 test document sets, each of which include approximately 10 news stories. Each document set is collected by a human and focuses on a particular topic. Example topics include the nomination of Clarence Thomas to the American Supreme Court, Neil Bush's role in the collapse of Silverado Savings and Loan and the Exxon Valdez oil spill. Gold standard summaries are provided for each document set for summary lengths of 50, 100, 200 and 400 words. This helps to ensure that the systems are not over-tuned to specific summary lengths. For each summary task (i.e., all 120 document set $\times$ summary length combinations), there are three distinct gold standard summaries created by different human analysts.

Pre-processing includes sentence boundary identification, segmentation of words (tokenisation), labelling words with part-of-speech tags, identification of noninflected base word forms (lemmatisation) from the LT-TTT tools (Grover et al., 2000). It also includes dependency parsing using Minipar (Lin, 1998) and automatic named entity recognition using the C&C tagger (Curran and Clark, 2003) trained on the data from the MUC-7 shared task (Chinchor, 1998). Weights for the various IE-based representations are calculated over each input document set.

5.2 Evaluation

The evaluation uses Rouge[9] to determine which representation selects content that overlaps most with human summaries. Rouge estimates the coverage of appropriate concepts (Lin, 2004) in a summary by comparing it to several human-created reference summaries. Rouge-1 does so by computing recall based on macro-averaged unigram overlap. Rouge-SU4 does so by calculating skip-bigram overlap where bigrams are allowed to be composed of non-contiguous words (with as many as four words intervening). Rouge-SU4 also includes unigrams to decrease the chances of zero scores where there is no skip-bigram overlap.

The configuration is based on comparisons between Rouge and human judgements of content coverage (Lin, 2004), which suggest that Rouge-1 and Rouge-SU4 with stemming and removal of stop words are good measures for evaluating multi-document summarisation tasks, consistently achieving Pearson's correlation scores above 0.72 and as high as 0.9 for longer summaries. Paired Wilcoxon signed ranks tests across document sets are used to check for significant differences between systems. The paired Wilcoxon signed ranks test is a non-parametric analogue of the paired t test. The null hypothesis is that the two populations from which the scores are sampled are identical.

1	50	100	200	400
TF	0.0797	0.1113	0.1742	0.2467
EV	**0.1360**	**0.1776**	0.2315	**0.3019**
RL	**0.1360**	0.1766	**0.2412**	0.3014

SU4	50	100	200	400
TF	0.0173	0.0259	0.0442	0.0693
EV	**0.0376**	**0.0494**	0.0692	**0.0950**
RL	0.0356	0.0491	**0.0701**	0.0939

Table 2: Comparison of Rouge scores for the *tf*idf* (*TF*), *event* (*EV*) and *relation* (*RL*).

6 Results

Can extractive summarisation be improved using representations based on generic information extraction? Table 2 contains results for *tf*idf* (*TF*), *event* (*EV*) and *relation* (*RL*) representations. Columns contain results for different lengths of summary (50, 100, 200 and 400 words). The best representation for each summary length is in bold and representations that are statistically distinguishable from the best (i.e., $p \leq 0.05$) are underlined. The results demonstrate unambiguously that the *event* and *relation* representations outperform the *tf*idf* representation, with strongly significant p-values less than 0.001 for both Rouge measures and all summary lengths. The *event* and *relation* representations are indistinguishable for both Rouge measures and all summary lengths.

[8] http://www-nlpir.nist.gov/projects/duc/index.html

[9] Rouge stands for recall-oriented understudy for gisting evaluations. While current versions also compute precision and f-score of system summaries, the evaluation here uses recall alone, which is sufficient when the length of the summaries being compared is the same. Rouge can be obtained from http://haydn.isi.edu/ROUGE/.

1	50	100	200	400
ER	**0.1497**	**0.1929**	**0.2527**	**0.3123**
EE	0.1442	0.1705	0.2288	0.3061

SU4	50	100	200	400
ER	**0.0419**	**0.0537**	**0.0786**	**0.1008**
EE	0.0364	0.0447	0.0643	0.0963

Table 3: Comparison of Rouge scores for entity pairs based on relations (*ER*) and events (*EE*).

How does entity pair identification for generic relations compare to entity pair identification for atomic events? Table 3 contains results for the representations described in Section 4.4. Rows correspond to entity pair identification for relations (*ER*) and events (*EE*).[10] Results suggest that the entity pair model based on GRE data outperforms the entity pair model based on atomic events, at least for medium sized summaries of 100 and 200 words where *ER* is significantly better than *EE* for both Rouge measures.

How do the event and relation representations perform with respect to corresponding entity pair representations? The scores for the entity pair representations reported in Table 3 are statistically indistinguishable from those for corresponding *relation* and *event* representations in Table 2 above. This appears to be a mixed result for both the *relation* representation introduced here and the Filatova and Hatzivassiloglou *event* representation. And, while GRE is shown to have a positive effect on Rouge scores when compared to atomic events, the same cannot be said of approaches to characterising relation and event types. However, as the correlation analysis (Section 7.1 below) demonstrates, RL and ER do not necessarily perform well on the same document sets. This suggests that they are actually complementary to some degree, meaning that a combined system based on both representations would outperform RL and ER on their own.

[10] In contrast to the results for the *tf*idf*, *relation* and *event* representations which use the modified adaptive algorithm described above, results for entity pair representations use a simplified version of the EXTRACT function that picks the text unit that has the highest score. This performed significantly better than the modified adaptive algorithm ($p \leq 0.01$) for all summary lengths for *ER* and was indistinguishable for *EE*. See Hachey (2009) for details.

7 Analysis and Comparison

7.1 Complementarity

Figure 4 contains results for a correlation analysis comparing the various representations. This also includes a comparison to the human upper bound (*HU*), computed by leave-one-out cross validation. Cells in the matrix contain the correlations values measured across document set Rouge-SU4 scores[11] using Spearman's ρ rank correlation coefficient (r_S). Here, high values mean that two representations tend to perform well on the same document sets such that an ordering of document sets by Rouge scores is similar for the representations being compared. In the figure, correlation strength is represented by shading where light-toned squares indicate strong correlation (and the darkest squares indicate weak negative correlation). For example, the upper left cell contains r_S between the TF and EV representations. The four squares correspond to r_S values of -0.085, 0.199, 0.245 and 0.267 respectively for summaries of 50, 100, 200 and 400 words.

The analysis illustrates a number of interesting points. First, it demonstrates that none of the representations correlate highly with the human upper bound, meaning that the automatic systems do not necessarily do well on the document sets that may be considered easier as measured by human agreement using Rouge. This suggests that task difficulty does not need to be considered as a possible underlying cause of correlation between the automatic systems. The analysis also illustrates that there is no clear and consistent relationship between summary length and correlation values. Some cells suggest that correlation may have a monotonic linear relationship increasing with length (e.g., TF*EV) while others seem to suggest inverse linear (e.g., TF*RL), quadratic (e.g., EV*HU) and invariant (e.g., EV*EE) relationships with length.

Looking at correlation between automatic systems (i.e., TF, EV, RL, EE and ER), correlation values closer to zero suggest that the systems do well on different document sets and that a combined system might therefore be better. By this reasoning, the largest gains would come from combining TF with any other representation. Among the other automatic systems, the *relation*

[11] Correlation across document set Rouge-1 scores shows similar trends.

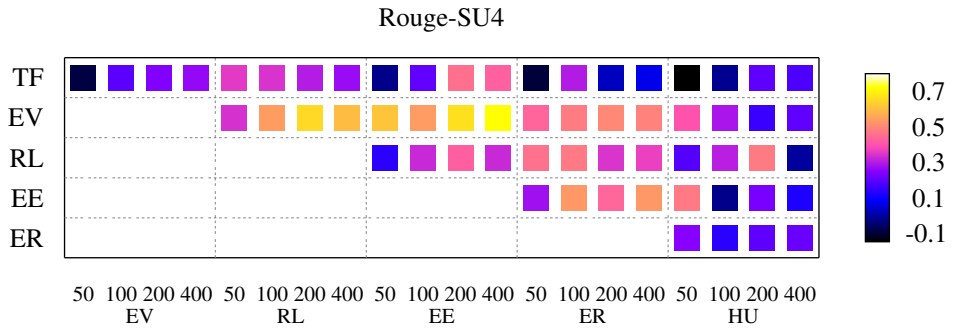


Figure 4: Comparison of representations using Spearman's r_s. Row and column labels correspond to *tf*idf* (*TF*), *event* (*EV*), *relation* (*RL*), *event entity pair* (*EE*), *relation entity pair* (*ER*) and human (*HU*) representations. Lighter toned squares indicate stronger correlation.

representation (RL) shows moderately high potential for combination with its corresponding entity pair representation (ER) with Spearman's r_S values in the range from 0.348 to 0.476. This suggests that ER should not necessarily be considered a simpler representation of the same information captured by RL when comparing results. The *event* representation (EV), by contrast, shows the strongest correlation of any comparison with its corresponding entity pair representation (EE) with r_S values in the range from 0.541 to 0.725.

7.2 Error Analysis

Four document sets were considered for error analysis. These were selected to cover different relative rankings of representations. Rows in Table 4 give the document set ID and list the representations in order of their Rouge-SU4 scores. Inspection of the corresponding document sets suggests that the different approaches compared here are appropriate for different types of summary tasks. Specifically, it suggests that *relation* and *event* representations perform poorly on summarisation tasks that are oriented towards sentiment, description or analysis. However, they do well on document sets that are oriented towards factual information typical of information extraction tasks (though current representations do not capture date, time or numeric information). This supports the notion from the previous section that the different representations evaluated here are complementary.

The document set (d06) for the summaries in Figure 5 illustrates a case where the *relation* and *event* representations perform well with respect

Set	Rank 1	Rank 2	Rank 3
d15	TF (0.046)	RL (0.035)	EV (0.023)
d39	TF (0.033)	EV (0.024)	RL (0.014)
d06	RL (0.094)	EV (0.060)	TF (0.016)
d53	RL (0.078)	TF (0.035)	EV (0.020)

Table 4: DUC 2001 document sets chosen for error analysis and corresponding Rouge-SU4 scores.

to *tf*idf*. The gold standard summary describes a beating event, addressing the basic facts of the Rodney King beating by Los Angeles police as well as the political aftermath which consists primarily of an investigation and a summary of related police brutality events. The difference in performance seems to be due to the fact that relations and events are central to all aspects of this summary and the *relation* and *event* representations clearly do better than *tf*idf* at capturing this information. This summary also illustrates an unintended side-effect of the *relation* representation where the generic relation identification algorithm finds relations between components of lexical compounds or multi-word phrases. The representation for the third sentence in the *RL* summary, for example, includes a relation between `ORG_police` and `XFN_chief` in addition to true positive relations e.g. between `ORG_police` and `PER_darylgates` and false positive relations e.g. between `PER_tombradley` and `ORG_police`.

7.3 Comparison to Supervised Extraction

Related work by Wong et al. (2008) also compares representations for sentence extraction on

TF (0.016) (20/29)	**[S1]** Mr. Williams likened the report to the Knapp Commission, a 1970s blue-ribbon study that exposed widespread corruption in the New York Police Department and led to significant improvements there. **[S2]** "There's no doubt in our mind that the only reason they stopped Joe Morgan was because he is black and he was the first black who happened to come by," said William Barnes, one of the attorneys representing the former ballplayer. **[S3]** Joseph McNamara, retired chief of San Jose's department and now a fellow at Stanford University's Hoover Institution, said he has been getting calls all summer from **[END]** cities around the country about racism and brutality in their departments.
EV (0.060) (9/29)	**[S1]** A high-ranking commission appointed after the beating, under the chairmanship of Mr Warren Christopher, a lawyer and former deputy secretary of state, concluded that the Los Angeles police department got results, in terms of arrests, but had developed a 'siege mentality that alienates the officer from the community'. **[S2]** The images of Los Angeles police swinging nightsticks at King as he lay on the ground, played repeatedly on national news programs, were burned into the national conscience and led to widespread calls for investigation of police brutality. **[S3]** Besides recommending that Mr Gates should go, the Christopher commission urged a policy **[END]** of community policing with more foot patrols, as well as measures to discipline racist police officers and to improve the investigation of complaints about police brutality.
RL (0.094) (3/29)	**[S1]** Mr. Gates opposed the Police Corps because its members would not be professionals. **[S2]** Shortly after Rodney King's beating, a news program on ABC illustrating police brutality showed a still photo of police using a martial-arts weapon against a person being arrested, but there was no mention that the episode involved Operation Rescue. **[S3]** The report was issued yesterday by a commission appointed by Mayor Tom Bradley and Police Chief Daryl Gates in the wake of the videotaped beating March 3 of a black motorist, Rodney King, by Los Angeles police. **[S4]** Investigations have been launched by the FBI, the Los **[END]** Angeles County district attorney's office and the Long Beach Police Department.
HU (0.400) (15/29)	The most important of the many cases of police brutality reported in southern California 1989-1992, was the beating of Rodney King by four Los Angeles officers on March 3, 1991. An investigating commission outlined steps for improvement of the police department and called for the resignation of Chief Gates. Gates did not resign until the following year after the acquittal of the four officers caused massive rioting. Other cases of police brutality arose in Minneapolis, Chicago and Kansas City. Operation Rescue claimed that its non-violent anti-abortion demonstrators were seriously injured by excessive police tactics in more than **[END]** 50 cities.

Figure 5: Example system and *human* (*HU*) summaries where *relation* (*RL*) and *event* (*EV*) representations perform well with respect to the *tf*idf* (*TF*) representation: Police Brutality Document Set (d06).

the DUC 2001 data. However, it uses supervised machine learning (probabilistic support vector machines) to derive a salience function while we focus on unsupervised approaches that can be ported to new domains and tasks without annotation or training. Interestingly, Wong et al.'s results suggest that adding events to a word-based feature set increases the precision of supervised sentence extraction but reduces the recall. By contrast, the current results and analysis provide evidence that word and generic IE-based representations are complementary when using unsupervised salience functions for sentence extraction.

The Wong et al. (2008) paper also provides useful results for comparison to state-of-the art. On the 200 word summarisation task, Wong et al. report Rouge-1 scores of 0.352 and 0.344 respectively for word-based and event-based representations. On the same task, our unsupervised approach achieves Rouge-1 scores of 0.174, 0.232, 0.229, 0.241 and 0.253 respectively for the *tf*idf*, *event*, *event entity pair*, *relation* and *relation entity pair* representations. Wong et al.'s best overall score is 0.396 using a representation that combines surface, content and relevance features.

8 Conclusion

Experiments were presented that compare the effect of various source document representations on the accuracy of automatic summarisation. This serves as an extrinsic evaluation of generic relation extraction, a domain-neutral and fully portable approach to relation identification and characterisation. Results demonstrate that GRE is an effective representation for sentence extraction for multi-document summarisation. Performance for the *relation* representation is significantly better than a non-trivial *tf*idf* baseline across the range of summary lengths explored. Performance is also at least as good as a comparable but less general representation based on event extraction. Correlation analysis suggests that different representations are complementary due to the fact that they perform well on different document sets. Error analysis supports this conclusion, suggesting that the *relation* and *event* representations perform poorly on summarisation tasks that are oriented towards e.g. sentiment, description or analysis while they perform well on tasks that focus on fact-oriented information.

Acknowledgments

This work was supported by Scottish Enterprise Edinburgh-Stanford Link grant R37588 as part of the EASIE project at the University of Edinburgh. It would not have been possible without the guidance of Claire Grover and Mirella Lapata.

References

Daniel Bienstock and Garud Iyengar. 2004. Faster approximation algorithms for packing and covering problems. Technical Report TR-2004-09, Columbia University.

David Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Nancy Chinchor. 1998. Overview of MUC-7. In *Proceedings of the 7th Message Understanding Conference*, Fairfax, VA, USA.

James R. Curran and Stephen Clark. 2003. Language independent NER using a maximum entropy tagger. In *Proceedings of the 7th Conference on Natural Language Learning*, Edmonton, Alberta, Canada.

Gerald DeJong. 1982. An overview of the FRUMP system. In Wendy G. Lehnert and Martin H. Ringle, editors, *Strategies for Natural Language Processing*, pages 149–176. Lawrence Erlbaum Associates, Hillsdale, NJ.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-based extractive summarization. In *Proceedings of the ACL Text Summarization Branches Out Workshop*, Barcelona, Spain.

Claire Grover, Colin Matheson, Andrei Mikheev, and Marc Moens. 2000. LT TTT—a flexible tokenisation tool. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece.

Ben Hachey. 2009. *Towards Generic Relation Extraction*. Ph.D. thesis, University of Edinburgh.

Udo Hahn and Ulrich Reimer. 1999. Knowledge-based text summarization: Salience and generalization operators for knowledge base abstraction. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*, pages 215–232. MIT Press, Cambridge, MA.

Sanda M. Harabagiu and Steven J. Maiorano. 2002. Multi-document summarization with GISTexter. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, Las Palmas, Spain.

Dorit S. Hochbaum. 1997. Approximating covering and packing problems: set cover, vertex cover, independent set and related problems. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 94–143. PWS Publishing Company, Boston, MA.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *Proceedings of the LREC Workshop Evaluation of Parsing Systems*, Granada, Spain.

Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Proceedings of the ACL Text Summarization Branches Out Workshop*, Barcelona, Spain.

Hans P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2).

Inderjeet Mani. 2001. *Automatic Summarization*. John Benjamins, Amsterdam/Philadelphia.

Kathleen R. McKeown, Desmond A. Jordan, and Vasileios Hatzivassiloglou. 1998. Generating patient-specific summaries of online literature. In *Proceedings of the AAAI Spring Symposium on Intelligent Text Summarization*, Stanford, CA, USA.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the 14th National Conference on Artificial Intelligence*, Portland, OR, USA.

Horacio Saggion and Guy Lapalme. 2002. Generating indicative-informative summaries with SumUM. *Computational Linguistics*, 28(4):497–526.

Karen Spärck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

Karen Spärck Jones. 2007. Automatic summarising: The state of the art. *Information Processing and Management*, 43:1449–1481.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles – experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–445.

Michael White and Claire Cardie. 2002. Selecting sentences for multidocument summaries using randomized local search. In *Proceedings of the ACL Workshop on Automatic Summarization*, Philadelphia, PA, USA.

Michael White, Tanya Korelsky, Claire Cardie, Vincent Ng, David Pierce, and Kiri Wagstaff. 2001. Multidocument summarization via information extraction. In *Proceedings of the 1st International Conference on Human Language Technology Research*, San Diego, CA, USA.

Kam-Fai Wong, Mingli Wu, and Wenjie Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd International Conference on Computational Linguistics*, Manchester, UK.

Language Models Based on Semantic Composition

Jeff Mitchell and **Mirella Lapata**
School of Informatics, University of Edinburgh
Edinburgh EH8 9LW, UK
jeff.mitchell@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In this paper we propose a novel statistical language model to capture long-range semantic dependencies. Specifically, we apply the concept of semantic composition to the problem of constructing predictive history representations for upcoming words. We also examine the influence of the underlying semantic space on the composition task by comparing spatial semantic representations against topic-based ones. The composition models yield reductions in perplexity when combined with a standard n-gram language model over the n-gram model alone. We also obtain perplexity reductions when integrating our models with a structured language model.

1 Introduction

Statistical language modeling plays an important role in many areas of natural language processing including speech recognition, machine translation, and information retrieval. The prototypical use of language models is to assign probabilities to sequences of words. By invoking the chain rule, these probabilities are generally estimated as the product of conditional probabilities $P(w_i|h_i)$ of a word w_i given the history of preceding words $h_i \equiv w_1^{i-1}$. In theory, the history could span any number of words up to w_i such as sentences or even a paragraphs. In practice, however, it has proven challenging to deal with the combinatorial growth in the number of possible histories which in turn impacts reliable parameter estimation. A simple and effective strategy is to truncate the chain rule to include only the n-1 preceding words (n is often set within the range of 3–5). The simplification reduces the number of free parameters. However, low values of n impose an artificially local horizon to the language model, and compromise its ability to capture long-range dependencies, such as syntactic relationships, semantic or thematic constraints.

The literature offers many examples of how to overcome this limitation, essentially by allowing the modulation of probabilities by dependencies which extend to words beyond the n-gram horizon. Cache language models (Kuhn and de Mori, 1992) increase the probability of words observed in the history, e.g., by some factor which decays exponentially with distance. Trigger models (Rosenfeld, 1996) go a step further by allowing arbitrary word pairs to be incorporated into the cache. Structured language models (e.g., Roark (2001)) go beyond the representation of history as a linear sequence of words to capture the syntactic constructions in which these words are embedded.

It is also possible to build representations of history which are semantic rather than syntactic (Bellegarda (2000; Coccaro and Jurafsky (1998; Gildea and Hofmann (1999)). In this approach, estimates for the probabilities of upcoming words are derived from a comparison of their semantic content with the content of the history so far. The semantic representations, in this case, are vectors derived from the distributional properties of words in a corpus, based on the insight that words which are semantically similar will be found in similar contexts (Harris, 1968; Firth, 1957). Although the the construction of a semantic representation for the history is crucial to this approach, the underlying vector-based models are primarily designed to represent isolated words rather than word sequences. Ideally, we would like to *compose* the meaning of the history out of its constituent parts. This is by no means a new idea. Much work in linguistic theory (Partee, 1995; Montague, 1974) has been devoted to compositionality, the process of determining the meaning of complex expressions from simpler ones. Previous work either ignores this issue (e.g., Bellegarda (2000)) or simply com-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 430–439,
Singapore, 6-7 August 2009.

putes the centroid of the vectors representing the history (e.g., Coccaro and Jurafsky (1998)). This is motivated primarily by mathematical convenience rather than by empirical evidence.

In our earlier work (Mitchell and Lapata, 2008) we formulated composition as a function of two vectors and introduced a variety of models based on addition and multiplication. In this paper we apply vector composition to the problem of constructing predictive history representations for language modeling. Besides integrating composition with language modeling, a task which is novel to our knowledge, our approach also serves as a valuable testbed of our earlier framework which we originally evaluated on a small scale verb-subject similarity task. We also investigate how the choice of the underlying semantic representation interacts with the choice of composition function by comparing a spatial model that represents words as vectors in a high-dimensional space against a probabilistic model that represents words as topic distributions.

Our results show that the proposed composition models yield reductions in perplexity when combined with a standard n-gram model over the n-gram model alone. We also show that with an appropriate composition function spatial models outperform the more sophisticated topic models. Finally, we obtain further perplexity reductions when our models are integrated with a structured language model, indicating that the two approaches to language modeling are complementary.

2 Background

2.1 Distributional Models of Semantics

The insight that words with similar meanings will tend to be distributed in similar contexts has given rise to a number of approaches that construct semantic representations from corpora. Broadly speaking, these models come in two flavors. *Semantic space* models represent the meaning of words in terms of vectors, with the vector components being derived from the distributional statistics of those words. Essentially, these models provide a simple procedure for constructing spatial representations of word meaning. *Topic models*, in contrast, impose a probabilistic model onto those distributional statistics, under the assumption that hidden topic variables drive the process that generates words. Both approaches represent the meanings of words in terms of an n-dimensional series of values, but whereas the semantic space model treats those values as defining a vector with spatial properties, the topic model treats them as a probability distribution.

A simple and popular (McDonald, 2000; Bullinaria and Levy, 2007; Lowe, 2000) way to construct a semantic space model is to associate each vector component with a particular context word, and assign it a value based on the strength of its co-occurrence with the target (i.e., the word for which a semantic representation is being constructed). For example, in Mitchell and Lapata (2008) we used the 2,000 most frequent content words in a corpus as their contexts, and defined co-occurrence in terms of the context word being present in a five word window on either side of the target word. We calculated the ratio of the probability of the context word given the target word to the overall probability of the context word and use these values as their vector components. This procedure has the benefits of simplicity and also of being largely free of any additional theoretical assumptions over and above the distributional approach to semantics. This is not to say that more sophisticated approaches have not been developed or that they are not useful. Much work has been devoted to enriching semantic space models with syntactic information (e.g., Grefenstette (1994; Padó and Lapata (2007)), selectional preferences (Erk and Padó, 2008) or with identifying optimal ways of defining the vector components (e.g., Bullinaria and Levy (2007)).

The semantic space discussed thus far is based on word co-occurrence statistics. However, the statistics of how words are distributed across the documents also carry useful semantic information. Latent Semantic Analysis (LSA, Landauer and Dumais (1997) utilizes precisely this distributional information to uncover hidden semantic factors by means of dimensionality reduction. Singular value decomposition (SVD, Berry et al. (1994)) is applied to a word-document co-occurrence matrix which is factored into a product of a number of other matrices; one of them represents words in terms of the semantic factors and another represents documents in terms of the same factors. The algebraic relation between these matrices can be used to show that any document vector is a linear combination of the vectors representing the words it contains. Thus, within this paradigm it is nat-

ural to treat multi-word structures as a "pseudo-document" and represent them via linear combinations of word vectors.

Due to its generality, LSA has proven a valuable analysis tool with a wide range of applications. However, the SVD procedure is somewhat ad-hoc lacking a sound statistical foundation. Probabilistic Latent Semantic Analysis (pLSA, Hofmann (2001)) casts the relationship between documents and words in terms of a generative model based on a set of hidden topics. Documents are represented by distributions over topics and topics are distributions over words. Thus the mixture of topics in any document determines its vocabulary. Maximum likelihood estimation of these distributions over a word-document matrix has a comparable effect to SVD in LSA: a set of hidden semantic factors, in this case topics, are extracted and documents and words are represented by these topics.

Latent Dirichlet Allocation (Griffiths et al., 2007; Blei et al., 2003) enhances further the mathematical foundation of this approach. Whereas pLSA treats each document as a separate, independent mixture of topics, LDA assumes that the topic distributions of documents are generated by a Dirichlet distribution. Thus, LDA is a probabilistic model of the whole document collection. In this model the process of generating a document can be described as follows:

1. draw a multinomial distribution θ from a Dirichlet distribution parametrized by α
2. for each word in a document:
   (a) draw a topic z_k from the multinomial distribution characterized by θ
   (b) draw a word from a multinomial distribution conditioned on the topic z_k and word probabilities β

Under this model, constructing a representation for a multi-word sequence amounts to estimating the topic proportions for that sequence.[1] Structure here arises from the mathematical form of the model, as opposed to any linguistic assumptions.

Without anticipating our results too much, we should point out that several features of the LDA model are likely to affect the representation of multi-word sequences. Firstly, it is a top-down generative model (the topic proportions for a document are first selected and then this drives the generation of words) as opposed to a bottom-up constructive process (words modulate each other to produce a complex representation of their combination). Secondly, the top level Dirichlet distribution is likely to lead to documents being dominated by a small number of topics, producing sparse vectors. And lastly, the assumption that words are generated independently means the interaction between them is not modeled.

[1] Estimating the posterior distribution $P(\theta, z|\mathbf{w}, \alpha, \beta)$ of the hidden variables given an observed collection of documents $\mathbf{w}$ is intractable in general; however, a variety of approximate inference algorithms have been proposed in the literature (e.g., Blei et al. (2003; Griffiths et al. (2007)).

2.2 Language Modeling using Semantic Representations

A common approach to embedding semantic representations within language modeling is to measure the semantic similarity between an upcoming word and its history and use it to modify the probabilities from an n-gram model. In this way, the n-gram's sensitivity to short-range dependencies is enriched with information about longer-range semantic coherence. Much of previous work has taken this approach (Bellegarda, 2000; Coccaro and Jurafsky, 1998; Wandmacher and Antoine, 2007), whilst relying on LSA to provide semantic representations for individual words. Some authors (Coccaro and Jurafsky, 1998; Wandmacher and Antoine, 2007) use the geometric notion of a vector centroid to construct representations of history, whereas others (Bellegarda, 2000; Deng and Khundanpur, 2003) use the idea of a "pseudo-document", which is derived from the algebraic relation between documents and words assumed within LSA. They all derive $P(w_i|h_i)$, the probability of an upcoming word given its history, from the cosine similarity measure which must be somehow normalized in order to yield well-formed probability estimates.

The approach of Gildea and Hofmann (1999) overcomes this difficulty by using representations constructed with pLSA, which have a direct probabilistic interpretation. As a result, the probability of an upcoming word given the history can be derived naturally and directly, avoiding the need for ad-hoc transformations. In constructing their representation of history, Gildea and Hofmann (1999) use an online Expectation Maximization process, which derives from the probabilistic basis of pLSA, to update the history with new words.

Extensions on the basic semantic language

models sketched above involve representing the history by multiple LSA models of varying granularity in an attempt to capture topic, subtopic, and local information (Zhang and Rudnicky, 2002); incorporating syntactic information by building the semantic space over words and their syntactic annotations (Kanejiya et al., 2004); and treating the LSA similarity as a feature in a maximum entropy language model (Deng and Khundanpur, 2003).

3 Composition Models

The problem of vector composition has received relatively little attention within natural language processing. Attempts to use tensor products (Smolensky, 1990; Clark et al., 2008; Widdows, 2008) as a means of binding one vector to another face major computational difficulties as their dimensionality grows exponentially with the number of constituents being composed. To overcome this problem, other techniques (Plate, 1995) have been proposed in which the binding of two vectors results in a vector which has the same dimensionality as its components. Crucially, the success of these methods depends on the assumption that the vector components are randomly distributed. This is problematic for modeling language which has regular structure.

Given the above considerations, in Mitchell and Lapata (2008) we introduce a general framework for studying vector composition, which we formulate as a function f of two vectors $\mathbf{u}$ and $\mathbf{v}$:

$$\mathbf{h} = f(\mathbf{u}, \mathbf{v}) \tag{1}$$

where $\mathbf{h}$ denotes the composition of $\mathbf{u}$ and $\mathbf{v}$. Different composition models arise, depending on how f is chosen. Our earlier work (Mitchell and Lapata, 2008) explored two broad classes of models based on additive and multiplicative functions.

Additive models are the most common method of vector combination in the literature. They have been applied to a wide variety of tasks including document coherence (Foltz et al., 1998), essay grading (Landauer and Dumais, 1997), modeling selectional restrictions (Kintsch, 2001), and notably language modeling (Coccaro and Jurafsky, 1998; Wandmacher and Antoine, 2007):

$$h_i = u_i + v_i \tag{2}$$

Vector addition (or averaging, which is equivalent under the cosine similarity measure) is a computationally efficient composition model as it does not increase the dimensionality of the resulting vector. However, the idea of averaging is somewhat counterintuitive from a linguistic perspective. Composition of simple elements onto more complex ones must allow the construction of novel meanings which go beyond those of the individual elements (Pinker, 1994).

In Mitchell and Lapata (2008) we argue that composition models based on multiplication address this problem:

$$h_i = u_i \cdot v_i \tag{3}$$

Whereas the addition of vectors 'lumps their content together', multiplication picks out the content relevant to their combination by scaling each component of one with the strength of the corresponding component of the other. This argument is appealing, especially if one is interested in explaining how the meaning of a verb is modulated by its subject. Here, we also develop a complementary, probabilistic argument for the validity of this model.

Let us assume that semantic vectors are based on components defined as the ratio of the conditional probability of a context word given the target word to the overall probability of the context word.

$$v_i = \frac{p(context_i|target)}{p(context_i)} \tag{4}$$

These vectors represent the distributional properties of a given target word in terms of the strength of its co-occurrence with a set of context words. Dividing through by the overall probability of each context word prevents the vectors being dominated by the most frequent context words, which will often also have the highest conditional probabilities.

Let us assume vectors $\mathbf{u}$ and $\mathbf{v}$ represent target words w_1 and w_2. Now, when we compose these vectors using the multiplicative model and the components definition in (4), we obtain:

$$h_i = v_i \cdot u_i = \frac{p(c_i|w_1)}{p(c_i)} \frac{p(c_i|w_2)}{p(c_i)} \tag{5}$$

And by Bayes' theorem:

$$h_i = \frac{p(w_1|c_i)p(w_2|c_i)}{p(w_1)p(w_2)} \tag{6}$$

Assuming w_1 and w_2 are independent and applying Bayes' theorem again, h_i becomes:

$$h_i \approx \frac{p(w_1 w_2|c_i)}{p(w_1 w_2)} = \frac{p(c_i|w_1 w_2)}{p(c_i)} \tag{7}$$

By comparing to (4), we can see that the expression on the right hand side gives us something akin to the vector components we would expect when our target is the co-occurrence of w_1 and w_2. Thus, for the multiplicative model, the combined vector h_i can be thought of as an approximation to a vector representing the distributional properties of the phrase $w_1 w_2$.

If multiplication results in a vector which is something like the representation of w_1 *and* w_2, then addition produces a vector which is more like the representation of w_1 *or* w_2. Suppose we were unsure whether a word token x was an instance of w_1 or of w_2. It would be reasonable to express the probabilities of context words around this token in terms of the probabilities for w_1 and w_2, assuming complete uncertainty between them:

$$p(c_i|x) = \frac{1}{2}p(c_i|w_1) + \frac{1}{2}p(c_i|w_2) \tag{8}$$

Therefore, we could represent x with a vector, based on these probabilities, having the components:

$$x_i = \frac{1}{2}\frac{p(c_i|w_1)}{p(c_i)} + \frac{1}{2}\frac{p(c_i|w_2)}{p(c_i)} \tag{9}$$

Which is exactly the vector averaging approach to semantic composition. As more vectors are combined, vector addition will lead to greater generality rather than greater specificity. The multiplicative approach, on the other hand, picks out the components of the constituents that are relevant to the combination, and represents more faithfully the properties of their conjunction.

As an aside, we should point out that our earlier work (Mitchell and Lapata, 2008) introduced several other models, additive and multiplicative, besides the ones discussed here. We selected the additive model as a baseline and also due to its overwhelming popularity in the language modeling literature. The multiplicative model presented above performed best in our evaluation study (i.e., predicting *verb-subject* similarity).

4 Language Modeling

Estimating Probabilities In language modeling our aim is to derive probabilities, $p(w|h)$, given the semantic representations of word, w, and its history, h, based on the assumption that probable words should be semantically coherent with the history. Semantic coherence is commonly measured via the cosine of the angle between two vectors:

$$sim(\mathbf{w},\mathbf{h}) = \frac{\mathbf{w}\cdot\mathbf{h}}{|\mathbf{w}||\mathbf{h}|} \tag{10}$$

$$\mathbf{w}\cdot\mathbf{h} = \sum_i w_i h_i \tag{11}$$

where $\mathbf{w}\cdot\mathbf{h}$ is the dot product of $\mathbf{w}$ and $\mathbf{h}$. Coccaro and Jurafsky (1998) utilize this measure in their approach to language modeling. Unfortunately, they find it necessary to resort to a number of ad-hoc mechanisms to turn the cosine similarities into useful probabilities. The primary problem with the cosine measure is that, although its values lie between 0 and 1, they do not sum to 1, as probabilities must. Thus, some form of normalization is required. A further problem concerns the fact that such a measure takes no account of the underlying frequency of w, which is crucial for a probabilistic model. For example, *encephalon* and *brain* are roughly synonymous, and may be equally similar to some context, but *brain* may nonetheless be much more likely, as it is generally more common.

An ideal measure would take account of the underlying probabilities of the elements involved and produce values that sum to 1. Our approach is to modify the dot product (equation (11)) on which the cosine measure is based. Assuming that our vector components are given by equation (4), the dot product becomes:

$$\mathbf{w}\cdot\mathbf{h} = \sum_i \frac{p(c_i|w)}{p(c_i)}\frac{p(c_i|h)}{p(c_i)} \tag{12}$$

which we modify to derive probabilities as follows:

$$p(w|h) = p(w)\sum_i \frac{p(c_i|w)}{p(c_i)}\frac{p(c_i|h)}{p(c_i)}p(c_i) \tag{13}$$

This expression now weights the sum with the independent probabilities of the context words and the word to be predicted. That this is indeed a valid probability can be seen by the fact it is equivalent to $\sum_i p(w|c_i)p(c_i|h)$. However, in constructing a representation of the history h, it is more convenient to work with equation (13) as it is based on vector components and can be readily used with the composition models presented in Mitchell and Lapata (2008).

Equation (13) allows us to derive probabilities from vectors representing a word and its prior history. We must also construct a representation of

the history up to the nth word of a sentence. To do this, we combine, via some (additive or multiplicative) function f, the vector representing that word with the vector representing the history up to $n-1$ words:

$$\mathbf{h}_n = f(\mathbf{w}_n, \mathbf{h}_{n-1}) \tag{14}$$

$$\mathbf{h}_1 = \mathbf{w}_1 \tag{15}$$

One issue that must be resolved in implementing equation (14) is that the history vector should remain correctly normalized. In other words, the products $h_i \cdot p(c_i)$ must themselves be a valid distribution over context words. So, after each vector composition the history vector is normalized as follows:

$$h_i = \frac{\hat{h}_i}{\sum_j \hat{h}_j \cdot p(c_i)} \tag{16}$$

Equations (13)–(16) define a language model that incorporates vector composition. To generate probability estimates, it requires a set of word vectors whose components are based on the ratio of probabilities described by equation (4).

Our discussion thus far has assumed a spatial semantic space model similar to that employed in Mitchell and Lapata (2008). However, there is no reason why the vectors should not be constructed by some other means. As mentioned earlier, in the LDA topic model, words are represented as distributions over topics. These distributions are essentially components of a vector $\mathbf{v}$ corresponding to the target word for which we wish to construct a semantic representation. Analogously to equation (4), we convert these probabilities to ratios of probabilities:

$$v_i = \frac{p(topic_i|target)}{p(topic_i)} \tag{17}$$

Integrating with Other Language Models The models defined above are based on little more than semantic coherence. As such they will be only weakly predictive, since they largely ignore word order, which n-gram models primarily exploit. The simplest means to integrate semantic information with a standard language model involves combining two probability estimates as a weighted sum:

$$p(w|h) = \lambda_1 p_1(w|h) + (1-\lambda) p_2(w|h) \tag{18}$$

Linear interpolation is guaranteed to produce valid probabilities, and has been used, for example, to integrate structured language models with n-gram models (Roark, 2001). However, it will work best when the models being combined are roughly equally predictive and have complementary strengths and weaknesses. If one model is much weaker than the other, linear interpolation will typically produce a model of intermediate strength (i.e., worse than the better model), with the weaker model contributing a form of smoothing at best.

Therefore, based on equation (13), we express our semantic probabilities as the product of the unigram probability, $p(w)$, and a semantic component, Δ, which determines the factor by which this probability should be scaled up or down given the context in which it occurs.

$$p(w|h) = p(w) \cdot \Delta(w,h) \tag{19}$$

$$\Delta(w,h) = \sum_i \frac{p(c_i|w)}{p(c_i)} \frac{p(c_i|h)}{p(c_i)} p(c_i) \tag{20}$$

Thus, it seems reasonable to integrate the n-gram model by replacing the unigram probabilities with the n-gram versions.[2]

$$\hat{p}(w_n) = p(w_n|w_{n-2}^{n-1}) \cdot \Delta(w_n, h) \tag{21}$$

To obtain a true probability estimate we normalize $\hat{p}(w_n)$ by dividing through the sum of all word probabilities:

$$p(w_n|w_{n-2}^{n-1}, h) = \frac{\hat{p}(w_n)}{\sum_w \hat{p}(w)} \tag{22}$$

In integrating our semantic model with an n-gram model, we allow the latter to handle short range dependencies and have the former handle the longer dependencies outside the n-gram window. For this reason, the history h used by the semantic model in the prediction of w_n only includes words up to w_{n-3} (i.e., only words outside the n-gram).

We also integrate our models with a structured language model (Roark, 2001). However, in this case we use linear interpolation (equation (18)) because the models are roughly equally predictive and also because linear interpolation is widely used when structured language models are combined with n-grams and other information sources. This approach also has the benefit of allowing the

[2] Equation (21) can also be expressed as $p(w_n|w_{n-2}^{n-1}, h) \approx \frac{p(w_n|w_{n-2}^{n-1}) p(w_n|h)}{p(w_n)}$, Which is equivalent to assuming that h is conditionally independent of w_{n-2}^{n-1} (Gildea and Hofmann, 1999).

models to be combined without out the need to renormalize the probabilities. In the case of the structured language model, normalizing across the whole vocabulary would be prohibitive.

5 Experimental Setup

In this section we discuss our experimental design for assessing the performance of the models presented above. We give details on our training procedure and parameter estimation, and present the methods used for comparison with our approach.

Method Following previous work (e.g., Bellegarda (2000)) we integrated our compositional language models with a standard n-gram model (see equation (21)). We experimented with additive and multiplicative composition functions, and two semantic representations (LDA and the simpler semantic space model), resulting in four compositional models. In addition, we compared our models against a state of the art structured language model in order to assess the extent to which the information provided by the semantic representation is complementary to syntactic structure. Our experiments used Roark's (2001) grammar-based language model. Similarly to standard language models, it computes the probability of the next word based upon the previous words of the sentence. This is done by computing a subset of all possible grammatical relations for the prior words and then estimating the probability of the next grammatical structure and the probability of seeing the next word given each of the prior grammatical relations. When estimating the probability of the next word, the model conditions on the two prior heads of constituents, thereby using information about word triples (like a trigram model).

All our models were evaluated by computing perplexity on the test set. Roughly, this quantifies the degree of unpredictability in a probability distribution, such that a fair k-sided dice would have a perplexity of k. More precisely, perplexity is the reciprocal of the geometric average of the word probabilities and a lower score indicates better predictions.

Parameter Estimation The compositional language models were trained on the BLLIP corpus, a collection of texts from the Wall Street Journal (years 1987–89). The training corpus consisted of 38,521,346 words. We used a development corpus of 50,006 words and a test corpus of similar size. All words were converted to lowercase and numbers were replaced with the symbol $\langle$num$\rangle$. A vocabulary of 20,000 words was chosen and the remaining tokens were replaced with $\langle$unk$\rangle$.

Following Mitchell and Lapata (2008), we constructed a simple semantic space based on co-occurrence statistics from the BLLIP training set. We used the 2,000 most frequent word types as contexts and a symmetric five word window. Vector components were defined as in equation (4). Contrary to our earlier work, we did not lemmatize the corpus before constructing the vectors as in the context of language modeling this was not appropriate. We also trained the LDA model on BLLIP, using Blei et al.'s (2003) implementation.[3] We experimented with different numbers of topics on the development set (from 10 to 200) and report results on the test set with 100 topics. In our experiments, the hyperparameter α was initialized to 0.5, and the β word probabilities were initialized randomly.

We integrated our compositional models with a trigram model which we also trained on BLLIP. The model was built using the SRILM toolkit (Stolcke, 2002) with backoff and Good-Turing smoothing. Ideally, we would have liked to train Roark's (2001) parser on the same data as that used for the semantic models. However, this would require a gold standard treebank several times larger than those currently available. Following previous work on structured language modeling (Roark, 2001; Charniak, 2001; Chelba and Jelinek, 1998), we therefore trained the parser on sections 2–21 of the Penn Treebank containing 936,017 words. Note that Roark's (2001) parser produces prefix probabilities for each word of a sentence which we converted to conditional probabilities by dividing each current probability by the previous one.

6 Results

Table 1 shows perplexity results when the compositional models are combined with an n-gram model. With regard to the simple semantic space model (SSM) we observe that both additive and multiplicative approaches to constructing history are successful in reducing perplexity over the n-gram baseline, with the multiplicative model outperforming the additive one. This confirms the

[3] Available from `http://www.cs.princeton.edu/~blei/lda-c/index.html`.

Model	Perplexity
n-gram	78.72
n-gram+Add$_{\text{SSM}}$	76.65
n-gram + Multiply$_{\text{SSM}}$	75.01
n-gram+Add$_{\text{LDA}}$	76.60
n-gram+Multiply$_{\text{LDA}}$	123.93
parser	173.35
n-gram + parser	75.22
n-gram + parser + Add$_{\text{SSM}}$	73.45
n-gram + parser + Multiply$_{\text{SSM}}$	71.32
n-gram + parser + Add$_{\text{LDA}}$	71.58
n-gram + parser + Multiply$_{\text{LDA}}$	87.93

Table 1: Perplexities for n-gram, composition and structured language models, and their combinations; subscripts $_{\text{SSM}}$ and $_{\text{LSA}}$ refer to the semantic space and LDA models, respectively.

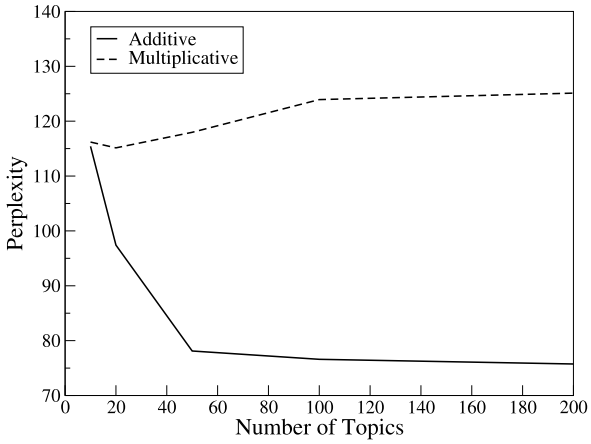


Figure 1: Perplexity versus Number of Topics for the LDA models using additive and multiplicative composition functions.

hypothesis that for this type of semantic space the multiplicative vector combination function produces representations which have a sounder probabilistic basis.

The results for the LDA model are also reported in the table. This model reduces perplexity with an additive composition function, but performs worse than the n-gram with a multiplicative function. For comparison, Figure 1 plots the perplexity of the combined LDA and n-gram models against the number of topics. Increasing the number of topics produces higher dimensional representations which ought to be richer, more detailed and therefore more predictive. While this is true for the additive model, a greater number of topics actually increases the perplexity of the multiplicative model, indicating it has become less predictive.

We compared these perplexity reductions against those obtained with a structured language model. Following Roark (2001), we combined the structured language model with a trigram model using linear interpolation (the weights were optimized on the development set). This model (n-gram + parser) performs comparably to our best compositional model (n-gram + Multiply$_{\text{SSM}}$). While both models incorporate long range dependencies, the parser is trained on a hand annotated treebank, whereas the compositional model uses raw text, albeit from a larger corpus. Interestingly, when interpolating the trigram with the parser *and* the compositional models, we obtain additional perplexity reductions. This suggests that the semantic models are encoding useful predictive information about long range dependencies, which is distinct from and potentially complementary to the parser's syntactic information about such dependencies. Note that the semantic space multiplicative model yields the highest perplexity reduction in this suite of experiments followed by the LDA additive model.

7 Conclusions

In this paper we advocated the use of vector composition models for language modeling. Using semantic representations of words outside the n-gram window, we enhanced a trigram model with longer range dependencies. We compared composition models based on addition and multiplication and examined the influence of the underlying semantic space on the composition task. Our results indicate that the multiplicative composition function produced the most predictive representations with a simple semantic space. Interestingly, its effect in the LDA setting was detrimental. Increasing the representational power of the LDA model, by using a greater number of topics, rendered the multiplicative model less predictive.

These results, together with the basic mathematical structure of the LDA model, suggest that it may not be well suited to forming representations for word sequences. In particular, the assumption that words are generated independently within documents prevents the interactions between words being modeled. This assumption, along with the Dirichlet prior on document distributions tends to lead to highly sparse word vec-

tors, with a typical word being strongly associated with only one or two topics. Multiplication of a number of these vectors generally produces a vector in which most of these associations have been obliterated by the sparse components, resulting in a representation with little predictive power.

These shortcomings arise from the mathematical formulation of LDA, which is not directed at modeling the semantic interaction between words. An interesting future direction would be to optimize the vector components of the probabilistic model over a suitable training corpus, in order to derive a vector model of semantics adapted specifically to the task of composition. We also plan to investigate more sophisticated composition models that take syntactic structure into account. Our results on interpolating the compositional models with a parser indicate that there is substantial mileage to be gained by combining syntactic and semantic dependencies.

Acknowledgements We are grateful to Brian Roark for making his parser available to us. Thanks to Frank Keller and Victor Lavrenko for insightful comments and suggestions. This work was supported by the Economic and Social Research Council [grant number PTA-030-2006-00341] and the Engineering and Physical Sciences Research Council [grant number GR/T04540/01].

References

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

Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. 1994. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

J.A. Bullinaria and J.P. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 116–123, Toulouse, France.

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, pages 225–231, Montréal, Canada.

Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. 2008. A compositional distributional model of meaning. In *Proceedings of the 2nd Symposium on Quantum Interaction*, pages 133–140, Oxford, UK. College Publications.

Noah Coccaro and Daniel Jurafsky. 1998. Towards better integration of semantic predictors in satistical language modeling. In *Proceedings of the 5th International Conference on Spoken Language Processing*, pages 2403–2406, Sydney, Australia.

Yonggang Deng and Sanjeev Khundanpur. 2003. Latent semantic information in maximum entropy language models for conversational speech recognition. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 56–63, Edmonton, AL.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 897–906, Honolulu, Hawaii.

J. R. Firth. 1957. A synopsis of linguistic theory 1930–1955. In *Studies in Linguistic Analysis*, pages 1–32. Philological Society, Oxford.

Peter Foltz, Walter Kintsch, and Thomas Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse Process*, 15:285–307.

Daniel Gildea and Thomas Hofmann. 1999. Topic-based language models using EM. In *Proceedings of the 6th European Conference on Speech Communiation and Technology*, pages 2167–2170, Budapest, Hungary.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Norwell, MA, USA.

Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114(2):211–244.

Zellig Harris. 1968. *Mathematical Structures of Language*. Wiley, New York.

Thomas Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 41(2):177–196.

Dharmendra Kanejiya, Arun Kumar, and Surendra Prasad. 2004. Statistical language modeling with performance benchmarks using various levels of

syntactic-semantic information. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 1161–1167, Geneva, Switzerland.

Walter Kintsch. 2001. Predication. *Cognitive Science*, 25(2):173–202.

Roland Kuhn and Renato de Mori. 1992. A cache based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (14):570–583.

T. K. Landauer and S. T. Dumais. 1997. A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240.

Will Lowe. 2000. *Topographic Maps of Semantic Space*. Ph.D. thesis, University of Edinburgh.

Scott McDonald. 2000. *Environmental Determinants of Lexical Processing Effort*. Ph.D. thesis, University of Edinburgh.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, OH.

R. Montague. 1974. English as a formal language. In R. Montague, editor, *Formal Philosophy*. Yale University Press, New Haven, CT.

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

B. Partee. 1995. Lexical semantics and compositionality. In Lila Gleitman and Mark Liberman, editors, *Invitation to Cognitive Science Part I: Language*, pages 311–360. MIT Press, Cambridge, MA.

S. Pinker. 1994. *The Language Instinct: How the Mind Creates Language*. HarperCollins, New York.

Tony A. Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Roni Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10:187–228.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:159–216.

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

Tonio Wandmacher and Jean-Yves Antoine. 2007. Methods to integrate a language model with semantic information for a word prediction component. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 506–513, Prague, Czech Republic.

Dominic Widdows. 2008. Semantic vector products: Some initial investigations. In *Proceedings of the 2nd Symposium on Quantum Interaction*, Oxford, UK. College Publications.

Rong Zhang and Alexander I. Rudnicky. 2002. Improve latent semantic analysis based language model by integrating multiple level knowldege. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 893–897, Denver, CO.

Graded Word Sense Assignment

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

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

Abstract

Word sense disambiguation is typically phrased as the task of labeling a word in context with the best-fitting sense from a sense inventory such as WordNet. While questions have often been raised over the choice of sense inventory, computational linguists have readily accepted the best-fitting sense methodology despite the fact that the case for discrete sense boundaries is widely disputed by lexical semantics researchers. This paper studies *graded word sense assignment*, based on a recent dataset of graded word sense annotation.

1 Introduction

The task of automatically characterizing word meaning in text is typically modeled as word sense disambiguation (WSD): given a list of senses for target lemma w, the task is to pick the best-fitting sense for a given occurrence of w. The list of senses is usually taken from an online dictionary or thesaurus. However, clear cut sense boundaries are sometimes hard to define, and the meaning of words depends strongly on the context in which they are used (Cruse, 2000; Hanks, 2000). Some researchers in lexical semantics have suggested that word meanings lie on a continuum between i) clear cut cases of ambiguity and ii) vagueness where clear cut boundaries do not hold (Tuggy, 1993). Certainly, it seems that a more complex representation of word sense is needed with a softer, graded representation of meaning rather than a fixed listing of senses (Cruse, 2000).

A recent annotation study ((Erk et al., 2009), hereafter **GWS**) marked a target word in context with graded ratings (on a scale of 1-5) on senses from WordNet (Fellbaum, 1998). Table 1 shows an example of a sentence with the target word in bold, and with the annotator judgments given to each sense. The study found that annotators made ample use of the intermediate ratings on the scale, and often gave high ratings to more than one WordNet sense for the same occurrence. It was found that the annotator ratings could not easily be transformed to categorial judgments by making more coarse-grained senses. If human word sense judgments are best viewed as graded, it makes sense to explore models of word sense that can predict graded sense assignments.

In this paper we look at the issue of graded applicability of word sense from the point of view of automatic *graded word sense assignment*, using the GWS graded word sense dataset. We make three primary contributions. Firstly, we propose evaluation metrics that can be used on graded word sense judgments. Some of these metrics, like Spearman's ρ, have been used previously (McCarthy et al., 2003; Mitchell and Lapata, 2008), but we also introduce new metrics based on the traditional precision and recall. Secondly, we investigate how two classes of models perform on the task of graded word sense assignment: on the one hand classical WSD models, on the other hand prototype-based vector space models that can be viewed as simple one-class classifiers. We study supervised models, training on traditional WSD data and evaluating against a graded scale. Thirdly, the evaluation metrics we use also provides a novel analysis of annotator performance on the GWS dataset.

2 Related Work

WSD has to date been a task where word senses are viewed as having clear cut boundaries. However, there are indications that word meanings do not behave in this way (Kilgarriff, 2006). Researchers in the field of WSD have acknowledged these problems but have used existing lexical resources in the hope that useful applications can be built with them. However, there is no consensus on which

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 440–449,
Singapore, 6-7 August 2009.

Sentence	Senses 1	2	3	4	5	6	7	Annotator
This can be justified thermodynamically in this case, and	2	3	3	5	5	2	3	Ann. 1
this will be done in a separate **paper** which is being	1	3	1	3	5	1	1	Ann. 2
prepared.	1	5	2	1	5	1	1	Ann. 3
	1.3	3.7	2	3	5	1.3	1.7	Avg

Table 1: A sample annotation in the GWS experiment. The senses are: 1 material from cellulose 2 report 3 publication 4 medium for writing 5 scientific 6 publishing firm 7 physical object

inventory is suitable for which application, other than cross-lingual applications where the inventory can be determined from parallel data (Carpuat and Wu, 2007; Chan et al., 2007). For monolingual applications however it is less clear whether current state-of-the-art WSD systems for tagging text with dictionary senses are able to have an impact on applications.

One way of addressing the problem of low inter-annotator agreement and system performance is to create an inventory that is coarse-grained enough for humans and computers to do the job reliably (Ide and Wilks, 2006; Hovy et al., 2006; Palmer et al., 2007). Such coarse-grained inventories can be produced manually from scratch (Hovy et al., 2006) or by automatically relating (McCarthy, 2006) or clustering (Navigli, 2006; Navigli et al., 2007) existing word senses. While the reduction in polysemy makes the task easier, we do not know which are the right distinctions to retain. In fact, fine-grained distinctions may be more useful than coarse-grained ones for some applications (Stokoe, 2005). Furthermore, Hanks (2000) goes further and argues that while the ability to distinguish coarse-grained senses is indeed desirable, subtler and more complex representations of word meaning are necessary for text understanding.

In this paper, instead of focusing on issues of granularity we try to predict graded judgments of word sense applicability, using a recent dataset with graded annotation (Erk et al., 2009). Our hope is that models which can mimic graded human judgments on the same task should better reflect the underlying phenomena of word meaning compared to a system that focuses on making clear cut distinctions. Also, we hope that such models might prove more useful in applications. There is one existing study of graded sense assignment (Ramakrishnan et al., 2004). It tries to estimate a probability distribution over senses by converting all of WordNet into a huge Bayesian Network, and reports improvements in a Question Answering task. However, it does not test its prediction against human annotator data.

We concentrate on supervised models in this paper since they generally perform better than their unsupervised or knowledge-based counterparts (Navigli, 2009). We compare them against a baseline model which simply uses the training data to obtain a probability distribution over senses regardless of context, since marginal distributions are highly skewed making a prior distribution very informative (Chan and Ng, 2005; Lapata and Brew, 2004).

Along with standard WSD models, we evaluate vector space models that use the training data to locate a word sense in semantic space. Word sense and vector space models have been related in two ways. On the one hand, vector space models have been used for inducing word senses (Schütze, 1998; Pantel and Lin, 2002). The different meanings of a word are obtained by clustering vectors. The clusters must then be mapped to an inventory if a standard WSD dataset is used for evaluation. In contrast, we use sense tagged training data with the aim of building models of given word senses, rather than clustering occurrences into word senses. The second way in which word sense and vector space models have been related is to assign disambiguated feature vectors to WordNet concepts (Pantel, 2005; Patwardhan and Pedersen, 2006). However those works do not use sense-tagged data and are not aimed at WSD, rather the applications are to insert new concepts into an ontology and to measure the relatedness of concepts.

We are not concerned in this paper with arguing for or against any particular sense inventory. WordNet has been criticized for being overly fine-grained (Navigli et al., 2007; Ide and Wilks, 2006), we are using it here because it is the sense inventory used by Erk et al. (2009). That annotation study used it because it is sufficiently fine-grained to allow for the examination of subtle distinctions between usages and because it is publicly available

lemma (PoS)	# senses	# training SemCor	SE-3
add (v)	6	171	238
argument (n)	7	14	195
ask (v)	7	386	236
different (a)	5	106	73
important (a)	5	125	11
interest (n)	7	111	160
paper (n)	7	46	207
win (v)	4	88	53
total training sentences		1047	1173

Table 2: Lemmas used in this study

with various sense-tagged datasets (e.g. (Miller et al., 1993; Mihalcea et al., 2004)) for comparison.

3 Data

In this paper, we use a subset of the GWS dataset (Erk et al., 2009) where three annotators supplied ordinal judgments of the applicability of WordNet (v3.0) senses on a 5 point scale: 1 – *completely different*, 2 – *mostly different*, 3 – *similar*, 4 – *very similar* and 5 – *identical*. Table 1 shows a sample annotation. The sentences that we use from the GWS dataset were originally extracted from the English SENSEVAL-3 lexical sample task (Mihalcea et al., 2004) (hereafter SE-3) and SemCor (Miller et al., 1993). [1] For 8 lemmas, 25 sentences were randomly sampled from SemCor and 25 randomly sampled from SE-3, giving a total of 50 sentences per lemma. The lemmas, their PoS and number of senses from WordNet are shown in table 2.

The annotation study found that annotators made ample use of the intermediate levels of applicability (2-4), and they often gave positive ratings (3-5) to more than one sense for a single occurrence. The example in Table 1 is one such case. An analysis of the annotator ratings found that they could not easily be explained in categorial terms by making more coarse-grained senses because senses that were not positively correlated often had high ratings for the same instance.

The GWS dataset contains a sequence of judgments for each occurrence of a target word in a sentence context: one judgment for each WordNet sense of the target word. To obtain a single judgment for each sense in each sentence we use the average judgment from the three annotators. As models typically assign values between 0 and 1, we normalize the annotator judgments from the GWS dataset to fall into the same range by using $normalized\_judgment = (judgment - 1.0)/4.0$. This maps an original judgment of 5 to a normalized judgment of 1.0, it maps an original 1 to 0.0, and intermediate judgments are mapped accordingly.

As the GWS dataset is too small to accommodate both training and testing of a supervised model, we use all the data from GWS for testing our models, and train our models on traditional word sense annotation data. We use as training data all sentences from SemCor and the training portion of SE-3 that are not included in GWS. The quantity of training data available is shown in the last two columns of table 2.

4 Evaluating Graded Word Sense Assignment

This section discusses measures for evaluating system performance for the case where gold judgments are graded rather than categorial.

Correlation. The standard method for comparing a list of graded gold judgments to a list of graded predicted judgments is by testing for correlation. In our case, as we cannot assume a normal distribution of the judgments, a non-parametric test such as Spearman's ρ will be appropriate. Spearman's ρ uses the formula of Pearson's coefficient, defined as

$$\rho(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y}$$

Pearson's coefficient computes the correlation of two random variables X and Y as their covariance divided by the product of their standard deviations. In the computation of Spearman's ρ, values are transformed to rankings before the formula is applied. [2] As Spearman's ρ compares the rankings of two sets of judgments, it abstracts from the absolute values of the judgments. It is useful to have a measure that abstracts from absolute values of judgments and magnitude of difference because the GWS dataset contains annotator judgments on a fixed scale, and it is quite possible that human judges will differ in how they use such a scale.

Each judgment in the gold-standard can be represented as a 4-tuple ⟨lemma, sense_no, sentence_no, gold_judgment⟩. For example, ⟨add.v,

[1] The GWS data also contains data from the English Lexical Substitution Task (McCarthy and Navigli, 2007) but we do not use that portion of the data for these experiments.

[2] Mitchell and Lapata (2008) note that Spearman's ρ tends to yield smaller coefficients than its parametric counterparts such as Pearson's coefficient.

1, 1, 0.8⟩ is the first sentence for target *add.v*, first WordNet sense, with a (normalized) judgment of 0.8. Likewise, each prediction by the model can be represented as a 4-tuple ⟨lemma, sense_no, sentence_no, predicted_judgment⟩. We write G for the set of gold tuples, A for the set of assigned tuples, L for the set of lemmas, S_ℓ for the set of sense numbers that exist for lemma ℓ, and T for the set of sentence numbers (there are 50 sentences for each lemma). We write $G|_{lemma=\ell}$ for the gold set restricted to those tuples with lemma ℓ, and analogously for other set restrictions and for A. There are several possibilities for measuring correlation:

by lemma: for each lemma $\ell \in L$, compute correlation between $G|_{lemma=\ell}$ and $A|_{lemma=\ell}$

by lemma+sense: for each lemma ℓ and each sense number $i \in S_\ell$, compute correlation between $G|_{lemma=\ell,senseno=i}$ and $A|_{lemma=\ell,senseno=i}$

by lemma+sentence: for each lemma ℓ and sentence number $t \in T$, compute correlation between $G|_{lemma=\ell,sentence=t}$ and $A|_{lemma=\ell,sentence=t}$

Comparison by lemma tests for the consistent use of judgments for the same target lemma. A comparison by lemma+sense ranks all occurrences of the same target lemma by how strongly they evoke a given word sense. A comparison by lemma+sentence ranks different senses by how strongly they apply to a given target lemma occurrence. In reporting correlation by lemma (by lemma+sense, by lemma+sentence), we average over all lemmas (lemma+sense, lemma+sentence combinations), and we report the percentage of lemmas (combinations) for which the correlation was significant. We report averaged correlation by lemma rather than one overall correlation over all judgments in order not to give more weight to lemmas with more senses.

Divergence. Another possibility for measuring the performance of a graded sense assignment model is to use Jensen/Shannon divergence (J/S), which is a symmetric version of Kullback/Leibler divergence. Given two probability distributions p, q, the Kullback/Leibler divergence of q from p is

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

and their J/S is

$$JS(p,q) = \frac{1}{2}\big(D(p||\frac{p+q}{2}) + D(q||\frac{p+q}{2})$$

We will use J/S for an evaluation by lemma+sentence: for each lemma $\ell \in L$ and sentence number $t \in T$, we normalize $G|_{lemma=\ell,sentence=t}$, the set of judgments for senses of ℓ in t, by the sum of sense judgments for ℓ and t. We do the same for $A|_{lemma=\ell,sentence=t}$. Then we compute J/S. In doing so, we are not trying to interpret $G|_{lemma=\ell,sentence=t}$ as some kind of probability distribution over senses, rather we use J/S as a measure that abstracts from absolute judgments but not from the magnitude of differences between judgments.

Precision and Recall. We have discussed a measure that abstracts from both absolute judgments and magnitude of differences (Spearman's ρ), and a measure that abstracts from absolute judgments but not the magnitude of differences (J/S). What is still missing is a measure that tests to what degree a model conforms to the absolute judgments given by the human annotators.

To obtain a measure for performance in predicting absolute gold judgments, we generalize precision and recall. In the categorial case, precision is defined as $P = \frac{\text{true positives}}{\text{true positives+false positives}}$, true positives divided by system-assigned positives, and recall is $R = \frac{\text{true positives}}{\text{true positives+false negatives}}$, true positives divided by gold positives. Writing $\text{gold}_{\ell,i,t}$ for the judgment j associated with lemma ℓ and sense number i for sentence t in the gold data (i.e., $\langle \ell, i, t, j\rangle \in G$), and analogously $\text{assigned}_{\ell,i,t}$, we extend precision and recall to the graded case as follows:

$$P_\ell = \frac{\sum_{i \in S_\ell, t \in T} \min(\text{gold}_{\ell,i,t}, \text{assigned}_{\ell,i,t})}{\sum_{i \in S_\ell, t \in T} \text{assigned}_{\ell,i,t}}$$

and

$$R_\ell = \frac{\sum_{i \in S_\ell, t \in T} \min(\text{gold}_{\ell,i,t}, \text{assigned}_{\ell,i,t})}{\sum_{i \in S_\ell, t \in T} \text{gold}_{\ell,i,t}}$$

where ℓ is a lemma. We compute precision and recall by lemma, then macro-average them in order not to give more weight to lemmas that have more senses. The formula for F-score as the harmonic mean of precision and recall remains unchanged: $F = 2\,P\,R/(P+R)$.

If the data is categorial, the graded precision and recall measures coincide with "classical" precision

Cx/2	until, IN, soft, JJ, remaining, VBG, ingredient, NNS
Cx/50	for, IN, sweet-sour, NN, sauce, NN, ..., to, TO, a, DT, boil, NN
Ch	OA, OA/ingredient/NNS

Table 3: Sample features for *add* in BNC occurrence *For sweet-sour sauce, cook onion in oil until soft.* ***Add*** *remaining ingredients and bring to a boil.* Cx/2 (Cx/50): context of size 2 (size 50) either side of the target. Ch: children of target.

and recall, which can be seen as follows. Graded sense assignment is represented by assigning each sense a score between 0.0 and 1.0. The categorial case can be represented in the same way, the difference being that one single sense will receive a score of 1.0 while all other senses get a score of 0.0. With this representation for categorial sense assignment, consider a fixed token t of lemma ℓ. $\sum_{i \in S_\ell} \min(\text{assigned}_{\ell,i,t}, \text{gold}_{\ell,i,t})$ will be 1 if the assigned sense is the gold sense, and 0 otherwise.

5 Models for Graded Word Sense Assignment

In this section we discuss the computational models for graded word sense that are tested in this paper.

Single-best-sense WSD. The first model that we test is a standard WSD model that assigns, to each test occurrence of a target word, a single best-fitting word sense. The system thus attributes a confidence score of 1 to the assigned sense and a confidence score of 0 for all other senses for that sentence. We refer to it as ***WSD/single***. The model uses standard features: lemma and part of speech in a narrow context window (2 words either side) and a wide context window (50 words either side), as well as dependency labels leading to parent, children, and siblings of the target word, and lemmas and part of speech of parent, child, and sibling nodes. Table 3 shows sample model features for an occurrence of *add* in the British National Corpus (BNC) (Leech, 1992). The model uses a maximum entropy learner[3], training one binary classifier per sense. (With n-ary classifiers, the model's performance is slightly worse.) The model is thus not highly optimized, but fairly standard.

WSD confidence level as judgment. Our second model is the same WSD system as above, but we use it to predict a judgment for each sense of a target occurrence, taking the confidence level returned by each sense-specific binary classifier as the predicted judgment. We refer to this model as ***WSD/conf***.

Word senses as points in semantic space. The results of the GWS annotation study raise the question of how word senses are best conceptualized, given that annotators assigned graded judgments of applicability of word senses, and given that they often combined high judgments for multiple word senses. One way of modeling these findings is to view word senses as prototypes, where some uses of a word will be typical examples of a given sense, for some uses the sense will clearly not apply, and to some uses the sense will be borderline applicable.

We use a very simple model of word senses as prototypes, representing them as points in a semantic space. Graded sense applicability judgments can then be modeled using vector similarity. The dimensions of the vector space are the features of the WSD system above (including dimensions like *Cx2/until*, *Cx2/IN*, *Ch/OA/ingredient/NNS* for the example in Table 3), and the coordinates are raw feature counts. We compute a single vector for each sense s, the centroid of all training occurrences that have been labeled with s. The predicted judgment for a test sentence and sense s is then the similarity of the sentence's vector to the centroid vector for s, computed using *cosine*. We call this model ***Prototype***. Like instance-based learners (Daelemans and den Bosch, 2005), the *Prototype* model measures the distance between feature vectors in space. Unlike instance-based learners, it only uses data from a single category for training.

As it is to be expected that the vectors in this space will be very sparse, we also test a variant of the *Prototype* model with Schütze-style second-order vectors (Schütze, 1998), called ***Prototype/2***. Given a (first-order) feature vector, we compute a second-order vector as the centroid of vectors for all lemma features (omitting stopwords) in the first-order vector. For the feature vector in Table 3, this is the centroid of vectors $\vec{sweet\text{-}sour}$, $\vec{sauce}$, ..., $\vec{boil}$. We compute the vectors $\vec{sweet\text{-}sour}$ etc. as dependency vectors (Padó and Lapata, 2007) [4] over a Minipar parse (Lin, 1993) of the BNC.

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

[4] We use the DV package, http://www.nlpado.de/~sebastian/dv.html, to compute the vector space.

We transform raw co-occurrence counts in the BNC-based vectors using pointwise mutual information (PMI), a common transformation function (Mitchell and Lapata, 2008). [5]

Another way of motivating the use of vector space models of word sense is by noting that we are trying to predict graded sense assignment by training on traditional word sense annotated data, where each target word occurrence is typically marked with a single word sense. Traditional word sense annotation, when used to predict GWS judgments, will contain spurious negative data: suppose a human annotator is annotating an occurrence of target word t and views senses s_1, s_2 and s_3 as somewhat applicable, with sense s_1 applying most clearly. Then if the annotation guidelines ask for the best-fitting sense, the annotator should only assign s_1. The occurrence is recorded as having sense s_1, but not senses s_2 and s_3. This, then, constitutes spurious negative data for senses s_2 and s_3. The simple vector space model of word sense that we use implements a radical solution to this problem of spurious negative data: it only uses positive data for a single sense, thus forgoing competition between categories. It is to be expected that not using competition between categories will hurt the vector space model's performance, but this design gives us the chance to compare two model classes that use opposing strategies with respect to spurious negative data: the WSD models fully trust the negative data, while the vector space models ignore it.

6 Experiments

This section reports on experiments for the task of graded word sense assignment. As data, we use the GWS dataset described in Sec. 3. We test the models discussed in Sec. 5, evaluating with the methods described in Sec. 4.

To put the models' performance into perspective, we first consider the human performance on the task, shown in Table 4. The first three lines of the table show the performance of each annotator evaluated against the average of the other two. The fourth line averages over the previous three lines to provide an average human ceiling for the task. In the correlation of rankings by lemma, correlation is statistically significant for all lemmas at $p \leq 0.01$. For correlation by lemma+sense and by lemma+sentence, the percentage of pairs with significant correlation is lower: 73.6 of lemma/sense pairs and 29.0 of lemma/sentence pairs reach significance at $p \leq 0.05$. For $p \leq 0.01$, the percentage is 58.3 and 12.2, respectively. The higher ρ but lower proportion of significant values for lemma+sentence pairs compared to lemma+sense is due to the fact that there are far fewer datapoints (sample size) for each calculation of ρ (#senses for lemma+sentence vs 50 sentences for lemma+sense).

At 0.131, J/S for Annotator 1 is considerably lower than for Annotators 2 and 3. [6] In terms of precision and recall, Annotator 1 again differs from the other two. At 87.5, her recall is higher than her precision (50.6), while the other annotators have considerably higher precision (75.5 and 82.4) than recall (62.4 and 52.3). This indicates that Annotator 1 tended to assign higher ratings throughout, an impression that is confirmed by Table 6. The left two columns show average ratings for each annotator over all senses of all tokens (normalized to values between 0.0 and 1.0 as described in Sec. 3). The three annotators differ widely in their average ratings, which range from 0.285 for Ann.3 to 0.540 for Ann.1.

Standard WSD. We tested the performance of the *WSD/single* model on a standard WSD task, using the same training and testing data as in our subsequent experiments, as described in section 3. [7] The model's accuracy when trained and tested on SemCor was A=77.0%, with a most frequent sense baseline of 63.5%. When trained and tested on SE-3, the model achieved A=53.0% against a baseline of 44.0%. When trained and tested on SemCor plus SE-3, the model reached an accuracy 58.2%, with a baseline of 56.0%. So on the combined dataset, the baseline is the average of the baselines on the individual datasets, while the model's performance falls below the average performance on the individual datasets.

WSD models for graded sense assignment. Table 5 shows the performance of different models in the task of graded word sense assignment. The first line in Table 5 lists results for the maximum entropy model when used to assign a single best sense. The second line lists the results for

[5] We also tested PMI transformation for the first-order vectors, but will not report the results here as they were worse across the board than without PMI.

[6] Low J/S implies a closer agreement between two sets of judgments.

[7] Note that this constitutes less training data than in the SE-3 task.

Ann	by lemma			by lemma+sense			by lemma+sentence			J/S	P	R	F
	ρ	*	**	ρ	*	**	ρ	*	**				
Ann.1	0.517	100.0	100.0	0.407	75.0	58.3	0.482	27.3	11.5	0.131	50.6	87.5	64.1
Ann.2	0.587	100.0	100.0	0.403	68.8	58.3	0.612	38.1	17.2	0.153	75.5	62.4	68.3
Ann.3	0.528	100.0	100.0	0.41	77.1	58.3	0.51	21.8	7.8	0.165	82.4	52.3	64.0
Avg	0.544	100.0	100.0	0.407	73.6	58.3	0.535	29.0	12.2	0.149	69.5	67.4	65.5

Table 4: Human ceiling: one annotator vs. average of the other two annotators. $*$, $**$: percentage significant at $p \leq 0.05$, $p \leq 0.01$. Avg: average annotator performance

Model	by lemma			by lemma+sense			by lemma+sentence			J/S	P	R	F
	ρ	*	**	ρ	*	**	ρ	*	**				
WSD/single	0.267	87.5	75.0	0.053	6.3	4.2	0.28	2.8	1.8	0.39	58.7	25.5	35.5
WSD/conf	0.396	87.5	87.5	0.177	33.3	18.8	0.401	10.8	3.0	0.164	81.8	37.1	51.0
Prototype	0.245	62.5	62.5	0.053	20.8	8.3	0.396	15.3	2.5	0.173	58.4	78.3	66.9
Prototype/2	0.292	87.5	87.5	0.086	14.6	4.2	0.478	22.8	7.5	0.164	68.2	63.3	65.7
Prototype/N	0.396	100.0	100.0	0.137	22.9	14.6	0.396	15.3	2.5	0.173	82.2	29.9	43.9
Prototype/2N	0.465	100.0	100.0	0.168	29.8	23.4	0.478	22.8	7.5	0.164	82.6	30.9	45.0
baseline	0.338	87.5	87.5	0.0	0.0	0.0	0.355	10.3	3.0	0.167	79.9	34.5	48.2

Table 5: Evaluation: computational models, and baseline. $*$, $**$: percentage significant at $p \leq 0.05$, $p \leq 0.01$

the same maximum entropy model when classifier confidence is used as predicted judgment. The last line shows the baseline, an adaptation of the most frequent sense baseline to the graded case. For this baseline, we computed the relative frequency of each sense in the training corpus and used this relative frequency as the prediction for each test sentence and sense combination. The *WSD/single* model remains below the baseline in all evaluations except correlation by lemma+sense, where no rank-based correlation could be computed for the baseline because it always assigns the same judgment for a given sense. *WSD/conf* shows a performance slightly above the baseline in all evaluation measures. Table 6 lists average ratings, averaged over all lemmas, senses, and occurrences, for each model in the two right-hand columns.

***Prototype* models.** Lines 3-6 in Tables 5 and 6 show results for *Prototype* variants. While each *Prototype* and *Prototype/2* model only sees positive data annotated for a single sense, the variants with */N* (lines 5 and 6) make very limited use of information coming from all senses of a given lemma. They normalize judgments for each sentence, with

$$\text{assigned}^{norm}_{\ell,i,t} = \frac{\text{assigned}_{\ell,i,t}}{\sum_{j \in S_\ell} \text{assigned}_{\ell,j,t}}$$

Line 3 evaluates the *Prototype* model with first-order vectors. Its correlation with the gold data is somewhat lower than that of *WSD/conf* in almost all cases. [8] The *Prototype* model deviates strongly from both *WSD/conf* and baseline in having a very good recall, at 78.3, with lower precision at 58.4, for an overall F-score that is 16 points higher than that of *WSD/conf*. Both *Prototype* and *Prototype/2* have average ratings (Table 6) far above those of the WSD models and of the */N* variants. The second-order vector model *Prototype/2* has relatively low correlation by lemma+sense, while correlation by lemma+sentence shows the best performance of all models (along with *Prototype/2N*). Its correlation by lemma+sentence is similar to the lowest correlation by lemma+sentence achieved by a human annotator. In terms of J/S, this model also shows the best performance along with *WSD/conf* and *Prototype/2N*. Both */N* variants achieve very high correlation by lemma. Correlation by lemma+sense for the */N* models is between those of *Prototype* and *WSD/conf*. The correlation by lemma+sentence is the same with or without normalization, as normalization does not change the ranking of senses of an individual sentence. While *Prototype* has higher recall than precision, normalization turns it into a model with even higher precision than *WSD/conf* but even lower recall.

Discussion

Human performance. The evaluation of human annotators in Table 4 provides a novel analysis of the GWS dataset over and above that by Erk et al.

[8] The reason why the average ρ for correlation by lemma+sense is the same for *Prototype* and *WSD/single* while the significance percentage differs greatly is that the *Prototype* shows negative correlation for some of the senses.

Ann.	avg	Model	avg
Ann.1	0.540	*WSD/single*	0.163
Ann.2	0.345	*WSD/conf*	0.173
Ann.3	0.285	*Prototype*	0.558
		Prototype/N	0.143
		Prototype/2	0.375
		Prototype/2N	0.143
		baseline	0.167

Table 6: Average judgment for individual annotators (transformed) and average rating for models

(2009). Human annotators show very strong correlation of their rankings by lemma. They also had strong agreement on rankings by lemma+sense, which ranks occurrences of a lemma by how strongly they evoke a given sense. The relatively low precision and recall in Table 4 confirm that different annotators use the 5-point scale in different ways. A comparison of precision and recall between the annotators reflects the fact that Annotator 1 tended to give considerably higher ratings than the other two, which is also apparent in the average ratings in Table 6. Given the relatively low F-score achieved by human annotators, judgments by additional annotators could make the GWS dataset more useful, in that the average judgments would not be influenced so strongly by idiosyncrasies in the use of the 5-point scale. (Psycholinguistic experiments using fixed scales typically elicit judgments from 10 or more participants per item.)

Evaluation measures. Given the degree of differences in the absolute values of the human annotator judgments (Table 4), a rank-based evaluation of graded sense assignment models, complemented by J/S to evaluate the magnitude of differences between ratings, seems most appropriate to the data. Rankings by lemma+sense and by lemma+sentence are especially interesting for their potential use in systems that might use graded sense assignment as part of a larger pipeline. Still, the new graded precision and recall measures allow for a more fine-grained analysis of the performance of models, showing fundamental differences in the behavior of *WSD/conf* and the *Prototype* model. Graded precision and recall could become even more informative measures with a gold set containing judgments of more annotators, since then the absolute gold judgments would be more reliable.

Standard WSD models and vector space models. The results in Table 5 reflect the compromise between the advantage of having competition between categories and the disadvantage of spurious negative data: *WSD/conf*, *Prototype/N* and *Prototype/2N* achieve the highest correlation by lemma, and high precision, while *Prototype* has much better recall for an overall higher F-score. However, as Table 6 shows, *Prototype* tends to assign high ratings across the board, leading to high recall. The much lower average ratings of the */N* models explain their higher precision and lower recall: they overshoot less and undershoot more. The improvement in correlation for the */N* models also indicates that *Prototype* assigns some sentences high ratings for all senses, impacting rankings by lemma and by lemma+sense.

The comparison of *Prototype* and *Prototype/2* gives us a chance to study effects of feature sparseness. *Prototype/2*, using second-order vectors that should be much less sparse, yields better rankings than *Prototype*. The average ratings of model *Prototype/2* (Table 6) are lower than those of *Prototype* (and closer to human average ratings), resulting in higher precision and lower recall. One possible reason for the high average ratings of *Prototype* is that in sparser (and shorter) vectors, matches in dimensions for high-frequency, relatively uninformative context items have greater impact.

It is interesting to see that *WSD/conf* performs slightly above the sense frequency baseline in all evaluations, since this is a very familiar picture from standard WSD.

Prototype/2N shows the overall most favorable performance in terms of correlation as it i) pays minimal attention to the negative data ii) uses normalization to avoid overshooting and iii) compensates for sparse data by using second order vectors. For J/S, *WSD/conf*, *Prototype/2*, *Prototype/2N* and the sense frequency baseline just outperform the score of the lowest-scoring of the three annotators. In terms of F-score, *Prototype* shows results very close to human performance. Interestingly, the *Prototype* model resembles Annotator 1 in its precision and recall, while *WSD/conf* more resembles Annotators 2 and 3. None of the models come close to human performance in ranking by lemma+sense, which requires an identification of the "typical" occurrence of a given sense. The low ratings in correlation by lemma+sense indicate that the models might be limited by the lack of training data for many of the rarer senses. In fu-

ture work, we will test how the frequency of senses in the training data affects the different models.

7 Conclusion

In this paper we have done a first study on modeling graded annotator judgments on sense applicability. We have discussed evaluation measures for models of graded sense assignment, including new extensions of precision and recall to the graded case. A combination of rank-based correlation at the level of lemmas, senses, and sentences, Jensen/Shannon divergence, and precision and recall provided a nuanced picture of the strengths and weaknesses of different models. We have tested two types of models: on the one hand a standard binary WSD model using classifier confidence as predicted judgments, and on the other hand several vector space models which compute a prototype vector for each sense in semantic space. These two types of model differ strongly in their behavior. The WSD model shows a similar behavior as the baseline, with high precision but low recall, while the unnormalized version of the vector space model has higher recall at lower precision. The results show both the benefits of having competition between categories, for improved rank-based correlation and precision, and the problem of spurious negative data in the training set arising from the best-sense methodology.

The last two correlation measures, by lemma+sense and by lemma+sentence, yield maybe the most insight into the question of the usability of a computational model for graded word sense assignment: a graded word sense assignment model that is a component of a larger system could provide useful sense information either by ranking occurrences by how strongly they evoke a sense, or by ranking senses by how strongly they apply to a given occurrence. There is room for improvement however as system performance is well below that of humans. In the future we plan to investigate features that are more informative for making graded judgments. Second, the vector space model we used was very simple; it might be worthwhile to test more sophisticated one-class classifiers (Marsland, 2003; Schölkopf et al., 2000).

Acknowledgments. We acknowledge support from the UK Royal Society for a Dorothy Hodgkin Fellowship to the second author.

References

M. Carpuat and D. Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proceedings of EMNLP-CoNLL 2007*, pages 61–72, Prague, Czech Republic, June. Association for Computational Linguistics.

Y. S. Chan and H. T. Ng. 2005. Word sense disambiguation with distribution estimation. In *Proceedings of IJCAI 2005*, pages 1010–1015, Edinburgh, Scotland.

Y. S. Chan, H. T. Ng, and D. Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of ACL'07*, Prague, Czech Republic, June.

D. A. Cruse. 2000. Aspects of the microstructure of word meanings. In Y. Ravin and C. Leacock, editors, *Polysemy: Theoretical and Computational Approaches*, pages 30–51. OUP, Oxford, UK.

W. Daelemans and A. Van den Bosch. 2005. *Memory-Based Language Processing*. Cambridge University Press, Cambridge, UK.

K. Erk, D. McCarthy, and N. Gaylord. 2009. Investigations on word senses and word usages. In *Proceedings of ACL-09*, Singapore.

C. Fellbaum, editor. 1998. *WordNet, An Electronic Lexical Database*. The MIT Press, Cambridge, MA.

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

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the HLT-NAACL 2006 workshop on Learning word meaning from non-linguistic data*, New York City, USA. Association for Computational Linguistics.

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

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

M. Lapata and C. Brew. 2004. Verb class disambiguation using informative priors. *Computational Linguistics*, 30(1):45–75.

G. Leech. 1992. 100 million words of English: the British National Corpus. *Language Research*, 28(1):1–13.

D. Lin. 1993. Principle-based parsing without overgeneration. In *Proceedings of ACL'93*, Columbus, Ohio, USA.

S. Marsland. 2003. Novelty detection in learning systems. *Neural computing surveys*, 3:157–195.

D. McCarthy and R. Navigli. 2007. SemEval-2007 task 10: English lexical substitution task. In *Proceedings of SemEval-2007*, pages 48–53, Prague, Czech Republic.

D. McCarthy, B. Keller, and J. Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL 03 Workshop: Multiword expressions: analysis, acquisition and treatment*, pages 73–80.

D. McCarthy. 2006. Relating WordNet senses for word sense disambiguation. In *Proceedings of the ACL Workshop on Making Sense of Sense: Bringing Psycholinguistics and Computational Linguistics Together*, pages 17–24, Trento, Italy.

R. Mihalcea, T. Chklovski, and A. Kilgarriff. 2004. The Senseval-3 English lexical sample task. In *Proceedings of SensEval-3*, Barcelona, Spain.

G. A. Miller, C. Leacock, R. Tengi, and R. T Bunker. 1993. A semantic concordance. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 303–308. Morgan Kaufman.

J. Mitchell and M. Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL'08 - HLT*, pages 236–244, Columbus, Ohio.

R. Navigli, K. C. Litkowski, and O. Hargraves. 2007. SemEval-2007 task 7: Coarse-grained English all-words task. In *Proceedings of SemEval-2007*, pages 30–35, Prague, Czech Republic.

R. Navigli. 2006. Meaningful clustering of senses helps boost word sense disambiguation performance. In *Proceedings of COLING-ACL 2006*, pages 105–112, Sydney, Australia.

R. Navigli. 2009. Word sense disambiguation: a survey. *ACM Computing Surveys*, 41(2):1–69.

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

M. Palmer, H. Trang Dang, and C. Fellbaum. 2007. Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Natural Language Engineering*, 13:137–163.

P. Pantel and D. Lin. 2002. Discovering word senses from text. In *Proceedings of KDD'02*.

P. Pantel. 2005. Inducing ontological co-occurrence vectors. In *Proceedings of ACL'05*, Ann Arbor, Michigan.

S. Patwardhan and T. Pedersen. 2006. Using wordnet-based context vectors to estimate the semantic relatedness of concepts. In *Proceedings of the EACL 06 Workshop: Making Sense of Sense: Bringing Psycholinguistics and Computational Linguistics Together*, Trento, Italy.

G. Ramakrishnan, B.P. Prithviraj, A. Deepa, P. Bhattacharyya, and S. Chakrabarti. 2004. Soft word sense disambiguation. In *Proceedings of GWC 04*, Brno, Czech Republic.

B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. 2000. Support vector method for novelty detection. *Advances in neural information processing systems*, 12.

H. Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1).

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

D. H. Tuggy. 1993. Ambiguity, polysemy and vagueness. *Cognitive linguistics*, 4(2):273–290.

Joint Learning of Preposition Senses and Semantic Roles of Prepositional Phrases

Daniel Dahlmeier[1], **Hwee Tou Ng**[1,2], **Tanja Schultz**[3]
[1]NUS Graduate School for Integrative Sciences and Engineering
[2]Department of Computer Science, National University of Singapore
[3]Cognitive Systems Lab, University of Karlsruhe
{danielhe,nght}@comp.nus.edu.sg
tanja@ira.uka.de

Abstract

The sense of a preposition is related to the semantics of its dominating prepositional phrase. Knowing the sense of a preposition could help to correctly classify the semantic role of the dominating prepositional phrase and vice versa. In this paper, we propose a joint probabilistic model for word sense disambiguation of prepositions and semantic role labeling of prepositional phrases. Our experiments on the PropBank corpus show that jointly learning the word sense and the semantic role leads to an improvement over state-of-the-art individual classifier models on the two tasks.

1 Introduction

Word sense disambiguation (WSD) and semantic role labeling (SRL) are two key components in natural language processing to find a semantic representation for a sentence. Semantic role labeling is the task of determining the constituents of a sentence that represent semantic arguments with respect to a predicate and labeling each with a semantic role. Word sense disambiguation tries to determine the correct meaning of a word in a given context. Ambiguous words occur frequently in normal English text.

One word class which is both frequent and highly ambiguous is preposition. The different senses of a preposition express different relations between the preposition complement and the rest of the sentence. Semantic roles and word senses offer two different inventories of "meaning" for prepositional phrases (PP): semantic roles distinguish between different verb complements while word senses intend to fully capture the preposition semantics at a more fine-grained level. In this paper, we use the semantic roles from the PropBank corpus and the preposition senses from the Preposition Project (TPP). Both corpora are explained in more detail in the following section. The relationship between the two inventories (PropBank semantic roles and TPP preposition senses) is not a simple one-to-one mapping, as we can see from the following examples:

- She now lives with relatives [in_{sense1} *Alabama.*]$_{\text{ARGM-LOC}}$
- The envelope arrives [in_{sense1} *the mail.*]$_{\text{ARG4}}$
- [In_{sense5} *separate statements*]$_{\text{ARGM-LOC}}$ the two sides said they want to have "further discussions."

In the first two examples, the sense of the preposition *in* is annotated as sense 1 ("surrounded by or enclosed in"), following the definitions of the TPP, but the semantic roles are different. In the first example the semantic role is a locative adjunctive argument (ARGM-LOC), while in the second example it is ARG4 which denotes the "end point or destination" of the arriving action[1]. In the first and third example, the semantic roles are the same, but the preposition senses are different, i.e., sense 1 and sense 5 ("inclusion or involvement").

Preposition senses and semantic roles provide two different views on the semantics of PPs. Knowing the semantic role of the PP could be helpful to successfully disambiguate the sense of the preposition. Likewise, the preposition sense could provide valuable information to classify the semantic role of the PP. This is especially so for the semantic roles ARGM-LOC and ARGM-TMP, where we expect a strong correlation with spatial and temporal preposition senses respectively.

In this paper, we propose a probabilistic model for joint inference on preposition senses and semantic roles. For each prepositional phrase that

[1]http://verbs.colorado.edu/framesets/arrive-v.html

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 450–458,
Singapore, 6-7 August 2009.

has been identified as an argument of the predicate, we jointly infer its semantic role and the sense of the preposition that is the lexical head of the prepositional phrase. That is, our model maximizes the *joint probability* of the semantic role and the preposition sense.

Previous research has shown the benefit of jointly learning semantic roles of *multiple constituents* (Toutanova et al., 2008; Koomen et al., 2005). In contrast, our joint model makes predictions for a *single constituent*, but *multiple tasks* (WSD and SRL) .

Our experiments show that adding the SRL information leads to statistically significant improvements over an independent, state-of-the-art WSD classifier. For the SRL task, we show statistically significant improvements of our joint model over an independent, state-of-the-art SRL classifier for locative and temporal adjunctive arguments, even though the overall improvement over all semantic roles is small. To the best of our knowledge, no previous research has attempted to perform preposition WSD and SRL of prepositional phrases in a joint learning approach.

The remainder of this paper is structured as follows: First, we give an introduction to the WSD and SRL task. Then, in Section 3, we describe the individual and joint classifier models. The details of the data set used in our experiments are given in Section 4. In Section 5, we present experiments and results. Section 6 summarizes related work, before we conclude in the final section.

2 Task Description

This section gives an introduction to preposition sense disambiguation and semantic role labeling of prepositional phrases.

2.1 Preposition Sense Disambiguation

The task of word sense disambiguation is to find the correct meaning of a word, given its context. Most prior research on word sense disambiguation has focused on disambiguating the senses of nouns, verbs, and adjectives, but not on prepositions. Word sense disambiguation can be framed as a classification task. For each preposition, a classifier is trained on a corpus of training examples annotated with preposition senses, and tested on a set of unseen test examples.

To perform WSD for prepositions, it is necessary to first find a set of suitable sense classes. We adopt the sense inventory from the Preposition Project (TPP) (Litkowski and Hargraves, 2005) that was also used in the SemEval 2007 preposition WSD task (Litkowski and Hargraves, 2007). TPP is an attempt to create a comprehensive lexical database of English prepositions that is suitable for use in computational linguistics research. For each of the over 300 prepositions and phrasal prepositions, the database contains a set of sense definitions, which are based on the Oxford Dictionary of English. Every preposition has a set of fine-grained senses, which are grouped together into a smaller number of coarse-grained senses. In our experiments, we only focus on coarse-grained senses since better inter-annotator agreement can be achieved on coarse-grained senses, which also results in higher accuracy of the trained WSD classifier.

2.2 Semantic Role Labeling

The task of semantic role labeling in the context of PropBank (Palmer et al., 2005) is to label tree nodes with semantic roles in a syntactic parse tree.

The PropBank corpus adds a semantic layer to parse trees from the Wall Street Journal section of the Penn Treebank II corpus (Marcus et al., 1993). There are two classes of semantic roles: core arguments and adjunctive arguments. Core arguments are verb sense specific, i.e., their meaning is defined relative to a specific verb sense. They are labeled with consecutive numbers ARG0, ARG1, etc. ARG0 usually denotes the AGENT and ARG1 the THEME of the event. Besides the core arguments, a verb can have a number of adjunctive arguments that express more general properties like time, location, or manner. They are labeled as ARGM plus a functional tag, e.g., LOC for locative or TMP for temporal modifiers. Prepositional phrases can appear as adjunctive arguments or core arguments.

The standard approach to semantic role labeling is to divide the task into two sequential sub-tasks: *identification* and *classification*. During the identification phase, the system separates the nodes that fill some semantic roles from the rest. During the classification phase, the system assigns the exact semantic roles for all nodes that are identified as arguments. In this paper, we focus on the classification phase. That is, we assume that prepositional phrases that are semantic arguments have been identified correctly and concentrate on the

task of determining the semantic role of prepositional phrases. The reason is that argument identification mostly relies on syntactic features, like the path from the constituent to the predicate (Pradhan et al., 2005). Consider, for example, the phrase *in the dark* in the sentence: "We are in the dark", he said. The phrase is clearly not an argument to the verb *say*. But if we alter the syntactic structure of the sentence appropriately (while the sense of the preposition *in* remains unchanged), the same phrase suddenly becomes an adjunctive argument: In the dark, he said "We are". On the other hand, we can easily find examples, where *in* has a different sense, but the phrase always fills some semantic role:

- In a separate manner, he said ...
- In 1998, he said ...
- In Washington, he said ...

This illustrates that the preposition sense is independent of whether the PP is an argument or not. Thus, a joint learning model for argument identification and preposition sense is unlikely to perform better than the independent models.

3 Models

This section describes the models for preposition sense disambiguation and semantic role labeling.

We compare three different models for each task: First, we implement an independent model that only uses task specific features from the literature. This serves as the *baseline model*. Second, we extend the baseline model by adding the most likely prediction of the other task as an additional feature. This is equivalent to a *pipeline model* of classifiers that feeds the prediction of one classification step into the next stage. Finally, we present a *joint model* to determine the preposition sense and semantic role that maximize the joint probability.

3.1 WSD model

Our approach to building a preposition WSD classifier follows that of Lee and Ng (2002), who evaluated a set of different knowledge sources and learning algorithms for WSD. However, in this paper we use maximum entropy models[2] (instead of support vector machines (SVM) reported in (Lee and Ng, 2002)), because maximum entropy models output probability distributions, unlike SVM. This property is useful in the joint model, as we will see later. Maxent models have been successfully applied to various NLP tasks and achieve state-of-the-art performance. There are two training parameters that have to be adjusted for maxent models: the number of training iterations and the Gaussian smoothing parameter. We find optimal values for both parameters through 10-fold cross-validation on the training set.

For every preposition, a baseline maxent model is trained using a set of features reported in the state-of-the-art WSD system of Lee and Ng (2002). These features encode three knowledge sources:

- Part-of-speech (POS) of surrounding words
- Single words in the surrounding context
- Local collocations

For *part-of-speech features*, we include the POS tags of surrounding tokens from the same sentence within a window of seven tokens around the target prepositions. All tokens (i.e., all words and punctuation symbols) are considered. We use the Penn Treebank II POS tag set.

For the knowledge source *single words in the surrounding context*, we consider all words from the same sentence. The input sentence is tokenized and all tokens that do not contain at least one alphabetical character (such as punctuation symbols and numbers) and all words that appear on a stopword list are removed. The remaining words are converted to lower case and replaced by their morphological root form. Every unique morphological root word contributes one binary feature, indicating whether or not the word is present in the context. The position of a word in the sentence is ignored in this knowledge source.

The third knowledge source, *local collocations*, encodes position-specific information of words within a small window around the target preposition. For this knowledge source, we consider unigrams, bigrams, and trigrams from a window of seven tokens. The position of the target preposition inside the n-gram is marked with a special character '_'. Words are converted to lower case, but no stemming or removal of stopwords is performed. If a token falls outside the sentence, it is replaced by the empty token symbol *nil*.

During testing, the maxent model computes the

[2]Zhang Le's Maximum Entropy Modeling Toolkit, http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

conditional probability of the sense, given the feature representation of the surrounding context c. The classifier outputs the sense that receives the highest probability:

$$\hat{s} = \underset{s}{\operatorname{argmax}}\, P(s|\Psi(c)) \quad (1)$$

where $\Psi(\cdot)$ is a feature map from the surrounding context to the feature representation.

To ensure that our model is competitive, we tested our system on the data set from the SemEval 2007 preposition WSD task (Litkowski and Hargraves, 2007). Our baseline classifier achieved a coarse-grained accuracy of 70.7% (micro-average) on the official test set. This would have made our system the second best system in the competition, behind the MELB-YB system (Ye and Baldwin, 2007).

We also investigate the effect of the semantic role label by adding it as a feature to the baseline model. This pipeline model is inspired by the work of Dang and Palmer (2005) who investigated the role of SRL features in verb WSD. We add the semantic role of the prepositional phrase dominating the preposition as a feature to the WSD model. During training, the PropBank gold SRL label is used. During testing, we rely on the baseline SRL model (to be introduced in the next subsection) to predict the semantic role of the prepositional phrase. This is equivalent to first performing semantic role labeling and adding the output as a feature to the WSD classifier. In earlier experiments, we found that training on gold SRL labels gave better results than training on automatically predicted SRL labels (using cross-validation). Note that our approach uses automatically assigned SRL labels during testing, while the system of Dang and Palmer (2005) only uses gold SRL labels.

3.2 SRL model

Our semantic role labeling classifier is also based on maxent models. It has been shown that maximum entropy models achieve state-of-the-art results on SRL (Xue and Palmer, 2004; Toutanova et al., 2008). Again, we find optimal values for the training parameters through 10-fold cross-validation on the training set.

By treating SRL as a classification problem, the choice of appropriate features becomes a key issue. Features are encoded as binary-valued functions. During testing, the maxent model computes the conditional probability $P(a|t, p, v)$ of the argument label a, given the parse tree t, predicate p, and constituent node v. The classifier outputs the semantic role with the highest probability:

$$\hat{a} = \underset{a}{\operatorname{argmax}}\, P(a|t, p, v) \quad (2)$$

$$= \underset{a}{\operatorname{argmax}}\, P(a|\Phi(t, p, v)) \quad (3)$$

where $\Phi(\cdot, \cdot, \cdot)$ is a feature map to an appropriate feature representation.

Baseline Features (Gildea and Jurafsky, 2002)	
pred	predicate lemma
path	path from constituent to predicate
ptype	syntactic category (NP, PP, etc.)
pos	relative position to the predicate
voice	active or passive voice
hw	syntactic head word of the phrase
sub-cat	rule expanding the predicate's parent
Advanced Features (Pradhan et al., 2005)	
hw POS	POS of the syntactic head word
PP hw/POS	head word and POS of the rightmost NP child if the phrase is a PP
first/last word	first/last word and POS in the constituent
parent ptype	syntactic category of the parent node
parent hw/POS	head word and POS of the parent
sister ptype	phrase type of left and right sister
sister hw/POS	head word and POS of left and right sister
temporal	temporal key words present
partPath	partial path predicate
proPath	projected path without directions
Feature Combinations (Xue and Palmer, 2004)	
pred & ptype	predicate and phrase type
pred & hw	predicate and head word
pred & path	predicate and path
pred & pos	predicate and relative position

Table 1: SRL features for the baseline model

For our baseline SRL model, we adopt the features used in other state-of-the-art SRL systems, which include the seven baseline features from the original work of Gildea and Jurafsky (2002), additional features taken from Pradhan *et al.* (2005), and feature combinations which are inspired by the system in Xue and Palmer (2004). Table 1 lists the features we use for easy reference.

In the pipeline model, we investigate the usefulness of the preposition sense as a feature for SRL by adding the preposition lemma concatenated with the sense number (e.g., *on_1*) as a feature. During training, the gold annotated preposition sense is used. During testing, the sense is automatically tagged by the baseline WSD model. This is equivalent to first running the WSD classifier for all prepositions, and adding the output preposition sense as a feature to our baseline SRL

system.

3.3 Joint Inference Model

The two previous models seek to maximize the probability of the semantic role and the preposition sense individually, thus ignoring possible dependencies between the two. Instead of maximizing the individual probabilities, we would like to maximize the *joint probability* of the semantic role and the preposition sense, given the parse tree, predicate, constituent node, and surrounding context.

$$\widehat{(a,s)} = \underset{(a,s)}{\operatorname{argmax}} P(a,s|t,p,v,c) \qquad (4)$$

We assume that the probability of the semantic role is already determined by the syntactic parse tree t, the predicate p, and the constituent node v, and is conditionally independent of the remaining surrounding context c given t, p, and v. Likewise, we assume that the probability of the preposition sense is conditionally independent of the parse tree t, predicate p, and constituent v, given the surrounding context c and the semantic role a. This assumption allows us to factor the joint probability into an SRL and a WSD component:

$$\widehat{(a,s)} = \underset{(a,s)}{\operatorname{argmax}} P(a|t,p,v) \times P(s|c,a) \qquad (5)$$

$$= \underset{(a,s)}{\operatorname{argmax}} P(a|\Phi(t,p,v)) \times P(s|\Psi(c,a)) \qquad (6)$$

We observe that the first component in our joint model corresponds to the baseline SRL model and the second component corresponds to the WSD pipeline model. Because our maxent models output a complete probability distribution, we can combine both components by multiplying the probabilities. Theoretically, the joint probability could be factored in the other way, by first computing the probability of the preposition sense and then conditioning the SRL model on the predicted preposition sense. However, in our early experiments, we found that this approach gave lower classification accuracy.

During testing, the classifier seeks to find the tuple of semantic role and preposition sense that maximizes the joint probability. For every semantic role, the classifier computes its probability given the SRL features, and multiplies it by the probability of the most likely preposition sense, given the context and the semantic role. The tuple that receives the highest joint probability is the final output of the joint classifier.

Semantic Role	Total	Training	Test
ARG0	28	15	13
ARG1	374	208	166
ARG2	649	352	297
ARG3	111	67	44
ARG4	177	91	86
ARGM-ADV	141	101	40
ARGM-CAU	31	23	8
ARGM-DIR	28	19	9
ARGM-DIS	29	9	20
ARGM-EXT	61	42	19
ARGM-LOC	954	668	286
ARGM-MNR	316	225	91
ARGM-PNC	115	78	37
ARGM-PRD	1	1	0
ARGM-REC	1	0	1
ARGM-TMP	838	563	275
Total	3854	2462	1392

Table 2: Number of annotated prepositional phrases for each semantic role

4 Data Set

The joint model uses the probability of a preposition sense, given the semantic role of the dominating prepositional phrase. To estimate this probability, we need a corpus which is annotated with both preposition senses and semantic roles. Unfortunately, PropBank is not annotated with preposition senses. Instead, we manually annotated the seven most frequent prepositions in four sections of the PropBank corpus with their senses from the TPP dictionary. According to Jurafsky and Martin (2008), the most frequent English prepositions are: *of*, *in*, *for*, *to*, *with*, *on* and *at* (in order of frequency). Our counts on Sections 2 to 21 of PropBank revealed that these top 7 prepositions account for about 65% of all prepositional phrases that are labeled with semantic roles.

The annotation proceeds in the following way. First, we automatically extract all sentences which have one of the prepositions as the lexical head of a prepositional phrase. The position of the preposition is marked in the sentence. By only considering prepositional phrases, we automatically exclude occurrences of the word *to* before infinitives and instances of particle usage of prepositions, such as phrasal verbs. The extracted prepositions are manually tagged with their senses from the TPP dictionary. Idiomatic usage of prepositions like *for example* or *in fact*, and complex preposition constructions that involve more than one word (e.g., *because of*, *instead of*, etc.) are excluded by the annotators and compiled into a stoplist.

We annotated 3854 instances of the top 7 prepo-

Preposition	Total	Training	Test
at	404	260	144
for	478	307	171
in	1590	1083	507
of	97	51	46
on	408	246	162
to	532	304	228
with	345	211	134
Total	3854	2462	1392

Table 3: Number of annotated prepositional phrases for each preposition

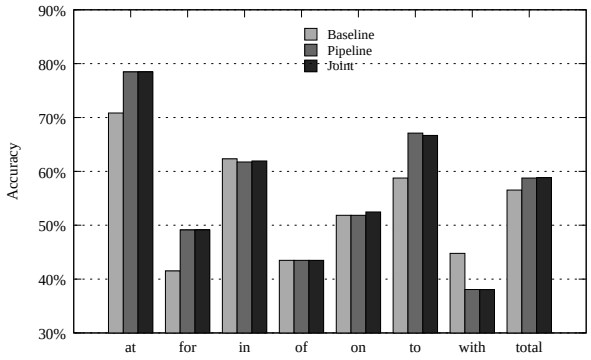


Figure 1: Classification accuracy of the WSD models for the seven most frequent prepositions in test section 23

sitions in Sections 2 to 4 and 23 of the PropBank corpus. The data shows a strong correlation between semantic roles and preposition senses that express a spatial or temporal meaning. For the preposition *in*, 90.8% of the instances that appear inside an ARGM-LOC are tagged with sense 1 ("surrounded by or enclosed in") or sense 5 ("inclusion or involvement"). 94.6% of the instances that appear inside an ARGM-TMP role are tagged with sense 2 ("period of time"). Our counts furthermore show that about one third of the annotated prepositional phrases fill core roles and that ARGM-LOC and ARGM-TMP are the most frequent roles. The detailed breakdown of semantic roles is shown in Table 2.

To see how consistent humans can perform the annotation task, we computed the inter-annotator agreement between two annotators on Section 4 of the PropBank corpus. We found that the two annotators assigned the same sense in 86% of the cases. Although not directly comparable, it is interesting to note that this figure is similar to inter-annotator agreement for open-class words reported in previous work (Palmer et al., 2000). In our final data set, all labels were tagged by the same annotator, which we believe makes our annotation reasonably consistent across different instances. Because we annotate running text, not all prepositions have the same number of annotated instances. The numbers for all seven prepositions are shown in Table 3. In our experiments, we use Sections 2 to 4 to train the models, and Section 23 is kept for testing. Although our experiments are limited to three sections of training data, it still allows us to train competitive SRL models. Pradhan *et al.* (2005) have shown that the benefit of using more training data diminishes after a few thousand training instances. We found that the accuracy of our SRL baseline model, which is trained on the 5275 sentences of these three sections, is only an absolute 3.89% lower than the accuracy of the same model when it is trained on twenty sections (71.71% accuracy compared to 75.60% accuracy).

5 Experiments and Results

We evaluate the performance of the joint model on the annotated prepositional phrases in test section 23 and compare the results with the performance of the baseline models and the pipeline models.

Figure 1 shows the classification accuracy of the WSD models for each of the seven prepositions in the test section. The results show that the pipeline model and the joint model perform almost equally, with the joint model performing marginally better in the overall score. The detailed scores are given in Table 4. Both models outperform the baseline classifier for three of the seven prepositions: *at*, *for*, and *to*. For the prepositions *in*, *of*, and *on*, the SRL feature did not affect the WSD classification accuracy significantly. For the preposition *with*, the classification accuracy even dropped by about 6%.

Performing the student's t-test, we found that the improvement for the prepositions *at*, *for*, and *to* is statistical significant ($p < 0.05$), as is the overall improvement. This confirms our hypothesis that the semantic role of the prepositional phrase is a strong hint for the preposition sense. However, our results also show that it is the SRL feature that brings the improvement, not the joint model, because the pipeline and joint model achieve about the same performance.

For the SRL task, we report the classification accuracy over all annotated prepositional phrases in the test section and the F_1 measure for the semantic roles ARGM-LOC and ARGM-TMP. Fig-

Preposition	Baseline	Pipeline	Joint
at	70.83	78.47*	78.47*
for	41.52	49.12*	49.12*
in	62.33	61.74	61.93
of	43.48	43.48	43.48
on	51.85	51.85	52.47
to	58.77	67.11*	66.67*
with	44.78	38.06	38.06
Total	56.54	58.76*	58.84*

Table 4: Classification accuracy of the baseline, pipeline, and joint model on the WSD task in test section 23, statistically significant improvements over the baseline are marked with an (*)

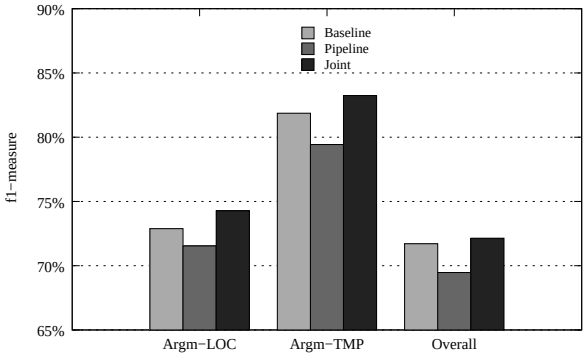


Figure 2: F_1 measure of the SRL models for ARGM-LOC and ARGM-TMP, and overall accuracy on prepositional phrases in test section 23

ure 2 shows the results. The joint model shows a small performance increase of 0.43% over the baseline in the overall accuracy. Adding the preposition sense as a feature, on the other hand, significantly lowers the accuracy by over 2%. For ARGM-LOC and ARGM-TMP, the joint model improves the F_1 measure by about 1.3% each. The improvement of the joint model for these roles is statistically significant ($p \leq 0.05$, student's t-test). Simply adding the preposition sense in the pipeline model again lowers the F_1 measure. The detailed results are listed in Table 5.

Semantic Role	Baseline	Pipeline	Joint
ARGM-LOC(F_1)	72.88	71.54	74.27*
ARGM-TMP(F_1)	81.87	79.43	83.24*
Overall(A)	71.71	69.47	72.14

Table 5: F_1 measure and accuracy of the baseline, pipeline, and joint model on the SRL task in test section 23, statistically significant improvements over the baseline are marked with an (*)

Our SRL experiments show that a pipeline model degrades the performance. The reason is the relatively high degree of noise in the WSD classification and that the pipeline model does not discriminate whether the previous classifier predicts the extra feature with high or low confidence. Instead, the model only passes on the 1-best WSD prediction, which can cause the next classifier to make a wrong classification based on the erroneous prediction of the previous step. In principle, this problem can be mitigated by training the pipeline model on automatically predicted labels using cross-validation, but in our case we found that automatically predicted WSD labels decreased the performance of the pipeline model even more. In contrast, the joint model computes the full probability distribution over the semantic roles and preposition senses. If the noise level in the first classification step is low, the joint model and the pipeline model perform almost identically, as we have seen in the previous WSD experiments. But if the noise level is high, the joint model can still improve while the pipeline model drops in performance. Our experiments show that the joint model is more robust in the presence of noisy features than the pipeline model.

6 Related Work

There is relatively less prior research on prepositions and prepositional phrases in the NLP community. O'Hara and Wiebe (2003) proposed a WSD system to disambiguate function tags of prepositional phrases. An extended version of their work was recently presented in (O'Hara and Wiebe, 2009). Ye and Baldwin (2006) extended their work to a semantic role tagger specifically for prepositional phrases. Their system first classifies the semantic roles of all prepositional phrases and later merges the output with a general SRL system. Ye and Baldwin (2007) used semantic role tags from surrounding tokens as part of the MELB-YB preposition WSD system. They found that the SRL features did not significantly help their classifier, which is different from our findings. Dang and Palmer (2005) showed that semantic role features are helpful to disambiguate verb senses. Their approach is similar to our pipeline WSD model, but they do not present results with automatically predicted semantic roles. Toutanova *et al.* (2008) presented a re-ranking model to jointly learn the semantic roles of multiple constituents in the SRL task. Their work dealt with joint learning in SRL, but it is not directly comparable to ours. The difference is that

Toutanova *et al.* attempt to jointly learn semantic role assignment of *different* constituents for *one* task (SRL), while we attempt to jointly learn *two tasks* (WSD and SRL) for *one* constituent. Because we only look at one constituent at a time, we do not have to restrict ourselves to a re-ranking approach like Toutanova *et al.*, but can calculate the full joint probability distribution of both tasks. Andrew *et al.* (2004) propose a method to learn a joint generative inference model from partially labeled data and apply their method to the problems of word sense disambiguation for verbs and determination of verb subcategorization frames. Their motivation is similar to ours, but they focus on learning from partially labeled data and they investigate different tasks.

None of these systems attempted to jointly learn the semantics of the prepositional phrase and the preposition in a single model, which is the main contribution of our work reported in this paper.

7 Conclusion

We propose a probabilistic model to jointly classify the semantic role of a prepositional phrase and the sense of the associated preposition. We show that learning both tasks together leads to an improvement over competitive, individual models for both subtasks. For the WSD task, we show that the SRL information improves the classification accuracy, although joint learning does not significantly outperform a simpler pipeline model here. For the SRL task, we show that the joint model improves over both the baseline model and the pipeline model, especially for temporal and location arguments. As we only disambiguate the seven most frequent prepositions, potentially more improvement could be gained by including more prepositions into our data set.

Acknowledgements

This research was supported by a research grant R-252-000-225-112 from National University of Singapore Academic Research Fund.

References

Galen Andrew, Trond Grenager, and Christopher D. Manning. 2004. Verb Sense and Subcategorization: Using Joint Inference to Improve Performance on Complementary Tasks. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 150–157.

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

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

Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized Inference with Multiple Semantic Role Labeling Systems. In *Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL 2005)*, pages 181–184.

Yoong Keok Lee and Hwee Tou Ng. 2002. An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 41–48.

Kenneth C. Litkowski and Orin Hargraves. 2005. The Preposition Project. In *Proceedings of the 2nd ACL-SIGSEM Workshop on The Linguistic Dimensions of Prepositions and Their Use in Computational Linguistic Formalisms and Applications*, pages 171–179.

Kenneth C. Litkowski and Orin Hargraves. 2007. SemEval-2007 Task 06: Word-Sense Disambiguation of Prepositions. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval 2007)*, pages 24–29.

Mitchell P. Marcus, Mary A. Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Tom O'Hara and Janyce Wiebe. 2003. Preposition Semantic Classification via Penn Treebank and FrameNet. In *Proceedings of the 7th Conference on Computational Natural Language Learning (CoNLL 2003)*, pages 79–86.

Tom O'Hara and Janyce Wiebe. 2009. Exploiting Semantic Role Resources for Preposition Disambiguation. *Computational Linguistics*, 35(2):151–184.

Martha Palmer, Hoa Trang Dang, and Joseph Rosenzweig. 2000. Sense Tagging the Penn Treebank. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000)*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–105.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Support Vector Learning for Semantic Argument Classification. *Machine Learning*, 60(1–3):11–39.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A Global Joint Model for Semantic Role Labeling. *Computational Linguistics*, 34(2):161–191.

Nianwen Xue and Martha Palmer. 2004. Calibrating Features for Semantic Role Labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 88–94.

Patrick Ye and Timothy Baldwin. 2006. Semantic Role Labeling of Prepositional Phrases. *ACM Transactions on Asian Language Information Processing (TALIP)*, 5(3):228–244.

Patrick Ye and Timothy Baldwin. 2007. MELB-YB: Preposition Sense Disambiguation Using Rich Semantic Features. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval 2007)*, pages 241–244.

Projecting Parameters for Multilingual Word Sense Disambiguation

Mitesh M. Khapra Sapan Shah Piyush Kedia Pushpak Bhattacharyya
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Powai, Mumbai – 400076,
Maharashtra, India.
{miteshk,sapan,charasi,pb}@cse.iitb.ac.in

Abstract

We report in this paper a way of doing Word Sense Disambiguation (WSD) that has its origin in multilingual MT and that is cognizant of the fact that *parallel corpora, wordnets* and *sense annotated corpora are scarce resources.* With respect to these resources, languages show different levels of readiness; *however a more resource fortunate language can help a less resource fortunate language*. Our WSD method can be applied to a language even when no sense tagged corpora for that language is available. This is achieved by ***projecting wordnet and corpus parameters*** from another language to the language in question. The approach is centered around a novel synset based multilingual dictionary and the empirical observation that within a domain the distribution of senses remains more or less invariant across languages. The effectiveness of our approach is verified by doing parameter projection and then running two different WSD algorithms. The accuracy values of approximately 75% (F1-score) for three languages in two different domains establish the fact that within a domain it is possible to circumvent the problem of scarcity of resources by projecting parameters like sense distributions, corpus-co-occurrences, conceptual distance, *etc.* from one language to another.

1 Introduction

Currently efforts are on in India to build large scale Machine Translation and Cross Lingual Search systems in consortia mode. These efforts are large, in the sense that 10-11 institutes and 6-7 languages spanning the length and breadth of the country are involved. The approach taken for translation is transfer based which needs to tackle the problem of word sense disambiguation (WSD) (Sergei *et. al*., 2003). Since 90s machine learning based approaches to WSD using sense marked corpora have gained ground (Eneko Agirre & Philip Edmonds, 2007). However, the creation of sense marked corpora has always remained a costly proposition. Statistical MT has obviated the need for elaborate resources for WSD, because WSD in SMT happens implicitly through parallel corpora (Brown *et. al*., 1993). But parallel corpora too are a very costly resource.

The above situation brings out the challenges involved in Indian language MT and CLIR. Lack of resources coupled with the multiplicity of Indian languages severely affects the performance of several NLP tasks. In the light of this, we focus on the problem of developing methodologies that *reuse resources*. The idea is to *do the annotation work for one language and find ways of using them for another language.*

Our work on WSD takes place in a multilingual setting involving *Hindi* (national language of India; 500 million speaker base), *Marathi* (20 million speaker base), *Bengali* (185 million speaker base) *and Tamil* (74 million speaker base). The wordnet of Hindi and sense marked corpora of Hindi are used *for all these languages.* Our methodology rests on a novel multilingual dictionary organization and on the idea of "parameter projection" from Hindi to the other languages. Also the domains of interest are *tourism* and *health.*

The roadmap of the paper is as follows. Section 2 describes related work. In section 3 we introduce the parameters essential for domain-specific WSD. Section 4 builds the case for parameter projection. Section 5 introduces the Multilingual Dictionary Framework which plays a key role in parameter projection. Section 6 is the core of the work, where we present parameter projection from one language to another. Section 7 describes two WSD algorithms which combine various parameters for do-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 459–467,
Singapore, 6-7 August 2009.

main-specific WSD. Experiments and results are presented in sections 8 and 9. Section 10 concludes the paper.

2 Related work

Knowledge based approaches to WSD such as Lesk's algorithm (Michael Lesk, 1986), Walker's algorithm (Walker D. & Amsler R., 1986), conceptual density (Agirre Eneko & German Rigau, 1996) and random walk algorithm (Mihalcea Rada, 2005) essentially do Machine Readable Dictionary lookup. However, these are fundamentally *overlap based* algorithms which suffer from overlap sparsity, dictionary definitions being generally small in length.

Supervised learning algorithms for WSD are mostly word specific classifiers, *e.g.*, WSD using SVM (Lee *et. al*., 2004), Exemplar based WSD (Ng Hwee T. & Hian B. Lee, 1996) and decision list based algorithm (Yarowsky, 1994). The requirement of a large training corpus renders these algorithms unsuitable for resource scarce languages.

Semi-supervised and unsupervised algorithms do not need large amount of annotated corpora, but are again word specific classifiers, *e.g.*, semi-supervised decision list algorithm (Yarowsky, 1995) and Hyperlex (Véronis Jean, 2004)). Hybrid approaches like WSD using Structural Semantic Interconnections (Roberto Navigli & Paolo Velardi, 2005) use combinations of more than one knowledge sources (wordnet as well as a small amount of tagged corpora). This allows them to capture important information encoded in wordnet (Fellbaum, 1998) as well as draw syntactic generalizations from minimally tagged corpora.

At this point we state that no single existing solution to WSD completely meets our requirements of **multilinguality, high domain accuracy** and **good performance in the face of not-so-large annotated corpora**.

3 Parameters for WSD

We discuss a number of parameters that play a crucial role in WSD. To appreciate this, consider the following example:

*The river flows through this region to meet the **sea.***

The word *sea* is ambiguous and has three senses as given in the Princeton Wordnet (PWN):

S1: (n) sea (a division of an ocean or a large body of salt water partially enclosed by land)
S2: (n) ocean, sea (anything apparently limitless in quantity or volume)
S3: (n) sea (turbulent water with swells of considerable size) "heavy seas"

Our first parameter is obtained from ***Domain specific sense distributions.*** In the above example, the first sense is more frequent in the tourism domain (verified from manually sense marked tourism corpora). Domain specific sense distribution information should be harnessed in the WSD task.

The second parameter arises from the ***dominance of senses in the domain.*** Senses are expressed by synsets, and we define a dominant sense as follows:

> A synset node in the wordnet hypernymy hierarchy is called *Dominant* if the synsets in the sub-tree below the synset are frequently occurring in the domain corpora.

A few dominant senses in the Tourism domain are *{place, country, city, area}*, *{body of water}*, *{flora, fauna}*, *{mode of transport}* and *{fine arts}*. In disambiguating a word, that sense which belongs to the sub-tree of a domain-specific dominant sense should be given a higher score than other senses. The value of this parameter (θ) is decided as follows:

θ = 1; if the candidate synset is a dominant synset

θ = 0.5; if the candidate synset belongs to the sub-tree of a dominant synset

θ = 0.001; if the candidate synset is neither a dominant synset nor belongs to the sub-tree of a dominant synset.

Our third parameter comes from ***Corpus co-occurrence.*** Co-occurring monosemous words as well as *already disambiguated words* in the context help in disambiguation. For example, the word *river* appearing in the context of *sea* is a monosemous word. The frequency of co-occurrence of *river* with the "water body" sense of *sea* is high in the tourism domain. Corpus co-occurrence is cal-

culated by considering the senses which occur in a window of 10 words around a sense.

Our fourth parameter is based on the ***semantic distance*** between any pair of synsets in terms of the shortest path length between two synsets in the wordnet graph. An edge in the shortest path can be any semantic relation from the wordnet relation repository (*e.g., hypernymy, hyponymy, meronymy, holonymy, troponymy etc.).*

For nouns we do something additional over and above the semantic distance. We take advantage of the deeper hierarchy of noun senses in the wordnet structure. This gives rise to our fifth and final parameter which arises out of the ***conceptual distance*** between a pair of senses. Conceptual distance between two synsets S_1 and S_2 is calculated using Equation (1), motivated by Agirre Eneko & German Rigau (1996).

$$\textit{Conceptual Distance (S1, S2)} = \frac{\text{Length of the path between (S1, S2) in terms of hypernymy hierarchy}}{\text{Height of the lowest common ancestor of S1 and S2 in the wordnet hierarchy}} \quad (1)$$

The conceptual distance is proportional to the path length between the synsets, as it should be. The distance is also inversely proportional to the height of the common ancestor of two sense nodes, because as the common ancestor becomes more and more general the conceptual relatedness tends to get vacuous (e.g., two nodes being related through *entity* which is the common ancestor of EVERYTHING, does not really say anything about the relatedness).

To summarize, our various parameters used for domain-specific WSD are:

Wordnet-dependent parameters

- *belongingness-to-dominant-concept*
- *conceptual-distance*
- s*emantic-distance*

Corpus-dependent parameters

- *sense distributions*
- *corpus co-occurrence*.

In section 7 we show how these parameters are used to come up with a scoring function for WSD.

4 Building a case for Parameter Projection

Wordnet-dependent parameters depend on the graph based structure of Wordnet whereas the ***Corpus-dependent parameters*** depend on various statistics learnt from a sense marked corpora. Both the tasks of (a) constructing a wordnet from scratch and (b) collecting sense marked corpora for multiple languages are tedious and expensive. An important question being addressed in this paper is: *whether the effort required in constructing semantic graphs for multiple wordnets and collecting sense marked corpora can be avoided?* Our findings seem to suggest that by ***projecting relations*** from the wordnet of a language and by ***projecting corpus*** statistics from the sense marked corpora of the language we can achieve this end. Before we proceed to discuss the way to realize parameter projection, we present a novel dictionary which facilitates this task.

5 Synset based multilingual dictionary

Parameter projection as described in section 4 rests on a novel and effective method of storage and use of dictionary in a multilingual setting proposed by Mohanty *et. al*. (2008). For the purpose of current discussion, we will call this multilingual dictionary framework *MultiDict*. One important departure from traditional dictionary is that **synsets are linked, and after that the words inside the synsets are linked.** The basic mapping is thus between synsets and thereafter between the words.

Concepts	**L1 (English)**	**L2 (Hindi)**	**L3 (Marathi)**
04321: a youthful male person	{male child, boy}	{लड़का ladkaa, बालक,baalak बच्चा bachchaa}	{मुलगाmulgaa , पोरगाporgaa , पोरpor }

Table 1: Multilingual Dictionary Framework

Table 1 shows the structure of MultiDict, with one example row standing for the concept of *boy*. The first column is the pivot describing a concept with a unique ID. The subsequent columns show the words expressing the concept in respective languages (in the example table above, *English, Hindi* and *Marathi*). Thus to express the concept *'04321: a youthful male person'*, there are two lexical elements in English, which constitute a *synset.* Correspondingly, the Hindi and Marathi synsets contain 3 words each.

It may be noted that the *central language* whose synsets the synsets of other languages link to is Hindi. This way of linking synsets- more popularly known as the *expansion* approach- has several advantages as discussed in (Mohanty *et. al.*, 2008). One advantage germane to the point of this paper is that the synsets in a particular column automatically inherit the various semantic relations of the Hindi wordnet (Dipak Narayan *et. al., 2000*), which saves the effort involved in reconstructing these relations for multiple languages.

After the synsets are linked, **cross linkages are set up** manually from the words of a synset to the words of a linked synset of the central language. The average number of such links per synset per language pair is approximately 3. These cross-linkages actually solve the problem of *lexical choice* in translating from text of one language to another.

Thus for the Marathi word मुलगा *{mulagaa}* denoting "*a youthful male person",* the correct lexical substitute from the corresponding Hindi synset is लड़का *{ladakaa}* (Figure 1). One might argue that any word within the synset could serve the purpose of translation. However, the exact lexical substitution has to respect native speaker acceptability.

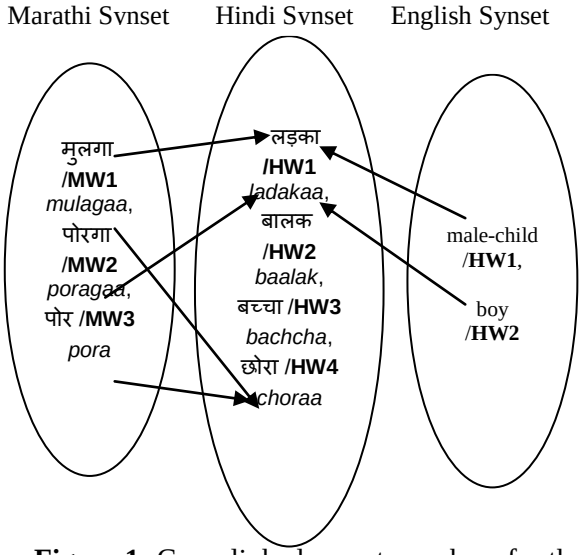


Figure 1: Cross linked synset members for the concept: *a youthful male person*

We put these cross linkages to another use, as described later.

Since it is the MultiDict which is at the heart of parameter projection, we would like to summarize the main points of this section. (1) By linking with the synsets of Hindi, the cost of building wordnets of other languages is partly reduced (semantic relations are inherited). The wordnet parameters of Hindi wordnet now become projectable to other languages. (2) By using the *cross linked words* in the synsets, corpus parameters become projectable (*vide* next section).

6 Parameter projection using MultDict

6.1 *P(Sense|Word)* parameter

Suppose a word (say, ***W***) in language L_1 (say, Marathi) has *k* senses. For each of these *k* senses we are interested in finding the parameter $P(S_i|W)$- which is the probability of sense S_i given the word *W* expressed as:

$$P(S_i|W) = \frac{\#(S_i, W)}{\sum_j \#(S_j, W)}$$

where '#' indicates 'count-of'. Consider the example of two senses of the Marathi word सागर *{saagar}, viz., sea* and *abundance* and the corresponding cross-linked words in Hindi (Figure 2 below):

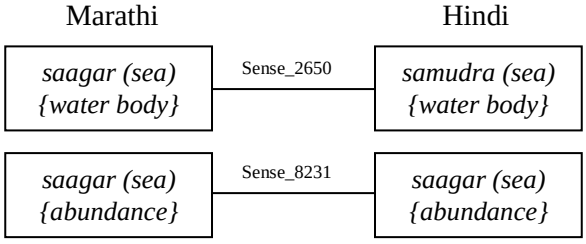


Figure 2: Two senses of the Marathi word सागर *(saagar), viz., {water body}* and *{abundance},* and the corresponding cross-linked words in Hindi[1].

The probability *P({water body}|saagar)* for Marathi is

$$\frac{\#(\{\boldsymbol{water\ body}\}, \boldsymbol{saagar})}{\#(\{\boldsymbol{water\ body}\}, \boldsymbol{saagar}) + \#(\{\boldsymbol{abundance}\}, \boldsymbol{saagar})}$$

We propose that this can be approximated by the counts from Hindi sense marked corpora by replacing *saagar* with the cross linked Hindi words *samudra* and *saagar*, as per Figure 2:

$$\frac{\#(\{\text{water body}\}, \text{samudra})}{\#(\{\text{water body}\}, \text{samudra}) + \#(\{\text{abundance}\}, \text{saagar})}$$

[1] Sense_8231 shows the same word *saagar* for both Marathi and Hindi. This is not uncommon, since Marathi and Hindi are sister languages.

Thus, the following formula is used for calculating the sense distributions of Marathi words using the sense marked Hindi corpus from the same domain:

$$P(S_i|W) = \frac{\#(S_i\ , cross\_linked\_hindi\_word)}{\sum_j\ \#(S_j\ , cross\_linked\_hindi\_word)} \qquad (2)$$

Note that we are not interested in the *exact* sense distribution of the words, but only in the *relative* sense distribution.

To prove that the projected relative distribution is faithful to the actual relative distribution of senses, we obtained the sense distribution statistics of a set of Marathi words from a sense tagged Marathi corpus (we call the sense marked corpora of a language its **self corpora**). These sense distribution statistics were compared with the statistics for these same words obtained by *projecting from* a sense tagged Hindi corpus using Equation (2). The results are summarized in Table 2.

Sr. No	Marathi Word	Synset	P(S|word) as learnt from sense tagged Marathi corpus	P(S|word) as projected from sense tagged Hindi corpus
1	किंमत (kimat)	{ worth }	0.684	0.714
		{ price }	0.315	0.285
2	रस्ता (rasta)	{ roadway }	0.164	0.209
		{road, route}	0.835	0.770
3	ठिकाण (thikan)	{ land site, place}	0.962	0.878
		{ home }	0.037	0.12
4	सागर (saagar)	{water body}	1.00	1.00
		{abundance}	0	0

Table 2: Comparison of the sense distributions of some Marathi words learnt from Marathi sense tagged corpus with those projected from Hindi sense tagged corpus.

The fourth row of Table 2 shows that whenever सागर *(saagar) (sea)* appears in the Marathi tourism corpus there is a 100% chance that it will appear in the *"water body"* sense and 0% chance that it will appear in the sense of *"abundance"*. Column 5 shows that the same probability values are obtained using projections from Hindi tourism corpus. Taking another example, the third row shows that whenever ठिकाण (*thikaan) (place, home)* appears in the Marathi tourism corpus there is a much higher chance of it appearing in the sense of "place" (96.2%) then in the sense of "home" (3.7%). Column 5 shows that the relative probabilities of the two senses remain the same even when using projections from Hindi tourism corpus (i.e. by using the corresponding cross-linked words in Hindi). To quantify these observations, we calculated the average KL divergence and Spearman's correlation co-efficient between the two distributions. The KL divergence is 0.766 and Spearman's correlation co-efficient is 0.299. Both these values indicate that there is a high degree of similarity between the distributions learnt using projection and those learnt from the self corpus.

6.2 Co-occurrence parameter

Similarly, within a domain, the statistics of co-occurrence of senses remain the same across languages. For example, the co-occurrence of the Marathi synsets {आकाश (akash) (sky), अंबर (ambar) (sky)} and {मेघ (megh) (cloud), अभ्र (abhra) (cloud)} in the Marathi corpus remains more or less same as (or proportional to) the co-occurrence between the corresponding Hindi synsets in the Hindi corpus.

Sr. No	Synset	Co-occurring Synset	P(co-occurrence) as learnt from sense tagged Marathi corpus	P(co-occurrence) as learnt from sense tagged Hindi corpus
1	{रोप, रोपटे} {small bush}	{झाड, वृक्ष, तरुवर, द्रुम, तरू, पादप} {tree}	0.125	0.125
2	{मेघ, अभ्र} {cloud}	{आकाश, आभाळ, अंबर} {sky}	0.167	0.154
3	{क्षेत्र, इलाक़ा, इलाका, भूखंड} {geographical area}	{यात्रा, सफ़र} {travel}	0.0019	0.0017

Table 3: Comparison of the corpus co-occurrence statistics learnt from Marathi and Hindi Tourism corpus.

Table 3 shows a few examples depicting similarity between co-occurrence statistics learnt from Marathi tourism corpus and Hindi tourism corpus. Note that we are talking about co-occurrence of synsets and not words. For example, the second row shows that the probability of co-occurrence of the synsets {cloud} and {sky} is almost same in the Marathi and Hindi corpus.

7 Our algorithms for WSD

We describe two algorithms to establish the usefulness of the idea of parameter projection. The first algorithm- called *iterative WSD (IWSD-)* is greedy, and the second based on PageRank algorithm is exhaustive. Both use scoring functions that make use of the parameters detailed in the previous sections.

7.1 Iterative WSD (IWSD)

We have been motivated by the Energy expression in Hopfield network (Hopfield, 1982) in formulating a scoring function for ranking the senses. Hopfield Network is a fully connected bidirectional symmetric network of bi-polar (0/1 or +1/-1) neurons. We consider the asynchronous Hopfield Network. At any instant, a randomly chosen neuron (a) examines the weighted sum of the input, (b) compares this value with a threshold and (c) gets to the state of 1 or 0, depending on whether the input is greater than or less than or equal to the threshold. The assembly of 0/1 states of individual neurons defines a state of the whole network. Each state has associated with it an energy, *E*, given by the following expression

$$E = -\theta_i V_i + \sum_{i=1}^{N} \sum_{j>i}^{N} W_{ij}\, V_i V_j \qquad (3)$$

where, *N* is the total number of neurons in the network, V_i and V_j are the activations of neurons *i* and *j* respectively and W_{ij} is the weight of the connection between neurons *i* and *j*. Energy is a fundamental property of Hopfield networks, providing the necessary machinery for discussing convergence, stability and such other considerations.

The energy expression as given above cleanly separates the influence of self-activations of neurons and that of interactions amongst neurons to the global macroscopic property of energy of the network. This fact has been the primary insight for equation (4) which was proposed to score the most appropriate synset in the given context. The correspondences are as follows:

Neuron	→	*Synset*
Self-activation	→	*Corpus Sense Distribution*
Weight of connection between two neurons	→	*Weight as a function of corpus co-occurrence and Wordnet distance measures between synsets*

$$S^* = \underset{i}{\operatorname{argmax}} \left(\theta_i * V_i + \sum_{j \in \mathrm{J}} W_{ij} * V_i * V_j \right) \quad (4)$$

where,

$\mathrm{J} = Set\ of\ disambiguated\ Words$

$\theta_i = BelongingnessToDominantConcept\ (S_i)$

$V_i = P(S_i \mid word)$

$W_{ij} = CorpusCooccurences\ (S_i, S_j) * 1/WNConceptualDistance(S_i, S_j) * 1/WNSemanticGraphDistance(S_i, S_j)$

The component $\theta_i * V_i$ of the energy due to the self activation of a neuron can be compared to the corpus specific sense of a word in a domain. The other component $w_{ij} * V_i * V_j$ coming from the interaction of activations can be compared to the score of a sense due to its interaction in the form of corpus co-occurrence, conceptual distance, and wordnet-based semantic distance with the senses of other words in the sentence. The first component thus captures the rather *static corpus sense*, whereas the second expression brings in the sentential context.

Algorithm 1: *performIterativeWSD(sentence)*

1. Tag all monosemous words in the sentence.
2. Iteratively disambiguate the remaining words in the sentence in increasing order of their degree of polysemy.
3. At each stage select that sense for a word which maximizes the score given by Equation (4)

Algorithm1: Iterative WSD

IWSD is clearly a greedy algorithm. It bases its decisions on already disambiguated words, and ignores words with higher degree of polysemy. For example, while disambiguating bisemous words, the algorithm uses only the monosemous words.

7.2 Modified PageRank algorithm

Rada Mihalcea (2005) proposed the idea of using PageRank algorithm to find the best combination of senses in a sense graph. The nodes in a sense graph correspond to the senses of all the words in a sentence and the edges depict the strength of interaction between senses. The score of each node in the graph is then calculated using the following recursive formula:

$$score(S_i) = (1-\mathrm{d}) + \mathrm{d} * \sum_{S_j \in In(S_i)} \frac{W_{ij}}{\sum_{S_k \in Out(S_i)} W_{jk}} * Score(S_j)$$

Instead of calculating W_{ij} based on the overlap between the definition of senses S_i and S_j as proposed by Rada Mihalcea (2005), we calculate the edge weights using the following formula:

$$W_{ij} = CorpusCooccurences(S_i, S_j) * 1/WNConceptualDistance(S_i, S_j) * 1/WNSemanticGraphDistance(S_i, S_j) * P(S_i \mid word_i) * P(S_j \mid word_j)$$

$d = damping\ factor\ (typically\ 0.85)$

This formula helps capture the edge weights in terms of the corpus bias as well as the interaction between the senses in the corpus and wordnet. It should be noted that this algorithm is *not greedy*. Unlike IWSD, this algorithm allows all the senses of all words to play a role in the disambiguation process.

8 Experimental Setup:

We tested our algorithm on tourism corpora in 3 languages (*viz.*, Marathi, Bengali and Tamil) and health corpora in 1 language (Marathi) using projections from Hindi. The corpora for both the domains were manually sense tagged. A 4-fold cross validation was done for all the languages in both the domains. The size of the corpus for each language is described in Table 4.

Language	**# of polysemous words (tokens)**	
	Tourism Domain	**Health Domain**
Hindi	50890	29631
Marathi	32694	8540
Bengali	9435	-
Tamil	17868	-

Table 4: Size of manually sense tagged corpora for different languages.

Table 5 shows the number of synsets in MultiDict for each language.

Language	**# of synsets in MultiDict**
Hindi	29833
Marathi	16600
Bengali	10732
Tamil	5727

Table 5: Number of synsets for each language

Algorithm	**Language**					
	Marathi			**Bengali**		
	P %	**R %**	**F %**	**P %**	**R %**	**F %**
IWSD (training on self corpora; no parameter projection)	81.29	80.42	80.85	81.62	78.75	79.94
IWSD (training on Hindi and reusing parameters for another language)	**73.45**	**70.33**	**71.86**	**79.83**	**79.65**	**79.79**
PageRank (training on self corpora; no parameter projection)	79.61	79.61	79.61	76.41	76.41	76.41
PageRank (training on Hindi and reusing parameters for another language)	**71.11**	**71.11**	**71.11**	**75.05**	**75.05**	**75.05**
Wordnet Baseline	58.07	58.07	58.07	52.25	52.25	52.25

Table 6: Precision, Recall and F-scores of IWSD, PageRank and Wordnet Baseline. Values are reported with and without parameter projection.

9 Results and Discussions

Table 6 shows the results of disambiguation (precision, recall and F-score). We give values for two algorithms in the tourism domain: IWSD and PageRank. In each case figures are given for both with and without parameter projection. The wordnet baseline figures too are presented for the sake of grounding the results.

Note the lines of numbers in bold, and compare them with the numbers in the preceding line. This shows the fall in accuracy value when one tries the parameter projection approach in place of self corpora. For example, consider the F-score as given by *IWSD* for Marathi. It degrades from about 81% to 72% in using parameter projection in place of self corpora. Still, the value is much more than the baseline, *viz.*, the wordnet first sense (a typically reported baseline).

Coming to PageRank for Marathi, the fall in accuracy is about 8%. Appendix A shows the corresponding figure for Tamil with IWSD as 10%. Appendix B reports the fall to be 11% for a different domain- Health- for Marathi (using *IWSD*).

In all these cases, even after degradation the performance is far above the wordnet baseline. This shows that one could trade accuracy with the cost of creating sense annotated corpora.

10 Conclusion and Future Work:

Based on our study for 3 languages and 2 domains, we conclude the following:

(i) Domain specific sense distributions- if obtainable- can be exploited to advantage.

(ii) Since sense distributions remain same across languages, it is possible to create a disambiguation engine that will work even in the absence of sense tagged corpus for some resource deprived language, provided (a) there are aligned and cross linked sense dictionaries for the language in question and another resource rich language, (b) the domain in which disambiguation needs to be performed for the resource deprived language is the same as the domain for which sense tagged corpora is available for the resource rich language.

(iii) Provided the accuracy reduction is not drastic, it may make sense to trade high accuracy for the effort in collecting sense marked corpora.

It would be interesting to test our algorithm on other domains and other languages to conclusively establish the effectiveness of parameter projection for multilingual WSD.

It would also be interesting to analyze the contribution of corpus and wordnet parameters independently.

References

Agirre Eneko & German Rigau. 1996. *Word sense disambiguation using conceptual density*. In Proceedings of the 16th International Conference on Computational Linguistics (COLING), Copenhagen, Denmark.

Dipak Narayan, Debasri Chakrabarti, Prabhakar Pande and P. Bhattacharyya. 2002. *An Experience in Building the Indo WordNet - a WordNet for Hindi*. First International Conference on Global WordNet, Mysore, India.

Eneko Agirre & Philip Edmonds. 2007. *Word Sense Disambiguation Algorithms and Applications*. Springer Publications.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database.* The MIT Press.

Hindi Wordnet.
http://www.cfilt.iitb.ac.in/wordnet/webhwn/

J. J. Hopfield. April 1982. *"Neural networks and physical systems with emergent collective computational abilities"*, Proceedings of the National Academy of Sciences of the USA, vol. 79 no. 8 pp. 2554-2558.

Lee Yoong K., Hwee T. Ng & Tee K. Chia. 2004. *Supervised word sense disambiguation with support vector machines and multiple knowledge sources*. Proceedings of Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Barcelona, Spain, 137-140.

Lin Dekang. 1997. *Using syntactic dependency as local context to resolve word sense ambiguity*. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL), Madrid, 64-71.

Michael Lesk. 1986. *Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone*. In Proceedings of the 5th annual international conference on Systems documentation, Toronto, Ontario, Canada.

Mihalcea Rada. 2005. *Large vocabulary unsupervised word sense disambiguation with graph-based algorithms for sequence data labeling*. In Proceedings of the Joint Human Language Technology and Empiri-

cal Methods in Natural Language Processing Conference (HLT/EMNLP), Vancouver, Canada, 411-418.

Ng Hwee T. & Hian B. Lee. 1996. *Integrating multiple knowledge sources to disambiguate word senses: An exemplar-based approach*. In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL), Santa Cruz, U.S.A., 40-47.

Peter F. Brown and Vincent J.Della Pietra and Stephen A. Della Pietra and Robert. L. Mercer. 1993. *The Mathematics of Statistical Machine Translation: Parameter Estimation*. Computational Linguistics Vol 19, 263-311.

Rajat Mohanty, Pushpak Bhattacharyya, Prabhakar Pande, Shraddha Kalele, Mitesh Khapra and Aditya Sharma. 2008. *Synset Based Multilingual Dictionary: Insights, Applications and Challenges*. Global Wordnet Conference, Szeged, Hungary, January 22-25.

Resnik Philip. 1997. *Selectional preference and sense disambiguation*. In Proceedings of ACL Workshop on Tagging Text with Lexical Semantics, Why, What and How? Washington, U.S.A., 52-57.

Roberto Navigli, Paolo Velardi. 2005. *Structural Semantic Interconnections: A Knowledge-Based Approach to Word Sense Disambiguation*. IEEE Transactions On Pattern Analysis and Machine Intelligence.

Sergei Nirenburg, Harold Somers, and Yorick Wilks. 2003. *Readings in Machine Translation.* Cambridge, MA: MIT Press.

Véronis Jean. 2004. *HyperLex: Lexical cartography for information retrieval*. Computer Speech & Language, 18(3):223-252.

Walker D. and Amsler R. 1986. *The Use of Machine Readable Dictionaries in Sublanguage Analysis*. In Analyzing Language in Restricted Domains, Grishman and Kittredge (eds), LEA Press, pp. 69-83.

Yarowsky David. 1994. *Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French*. In Proceedings of the 32nd Annual Meeting of the association for Computational Linguistics (ACL), Las Cruces, U.S.A., 88-95.

Yarowsky David. 1995. *Unsupervised word sense disambiguation rivaling supervised methods*. In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL), Cambridge, MA, 189-196.

Appendix A: Results for Tamil (Tourism Domain)

Algorithm	P %	R %	F %
IWSD (training on Tamil)	89.50	88.18	88.83
IWSD (training on Hindi and reusing for Tamil)	84.60	73.79	78.82
Wordnet Baseline	65.62	65.62	65.62

Table 7: Tamil Tourism corpus using parameters projected from Hindi

Appendix B: Results for Marathi (Health Domain)

Algorithm Words	P %	R %	F %
IWSD (training on Marathi)	84.28	81.25	82.74
IWSD (training on Hindi and reusing for Marathi)	75.96	67.75	71.62
Wordnet Baseline	60.32	60.32	60.32

Table 8: Marathi Health corpus parameters projected from Hindi

Multi-Word Expression Identification Using Sentence Surface Features

Ram Boukobza
School of Computer Science
Hebrew University of Jerusalem
ram.boukobza@mail.huji.ac.il

Ari Rappoport
School of Computer Science
Hebrew University of Jerusalem
arir@cs.huji.ac.il

Abstract

Much NLP research on Multi-Word Expressions (MWEs) focuses on the discovery of new expressions, as opposed to the identification in texts of known expressions. However, MWE identification is not trivial because many expressions allow variation in form and differ in the range of variations they allow. We show that simple rule-based baselines do not perform identification satisfactorily, and present a supervised learning method for identification that uses sentence surface features based on expressions' canonical form. To evaluate the method, we have annotated 3350 sentences from the British National Corpus, containing potential uses of 24 verbal MWEs. The method achieves an F-score of 94.86%, compared with 80.70% for the leading rule-based baseline. Our method is easily applicable to any expression type. Experiments in previous research have been limited to the compositional/non-compositional distinction, while we also test on sentences in which the words comprising the MWE appear but not as an expression.

1 Introduction

Multi-Word Expressions (MWEs) such as 'pull strings', 'make a face' and 'get on one's nerves' are very common in language. Such MWEs can be characterized as being *non-compositional*: the meaning of the expression does not transparently follow from the meaning of the words that comprise it. Much of the work on MWEs in NLP has been in *MWE extraction* – the discovery of new MWEs from a corpus, using statistical and other methods. Identification of *known* MWEs in text has received less attention, but is necessary for many NLP applications, for example in machine translation. The current work deals with the *MWE identification* task: deciding if a sentence contains a use of a known expression.

MWE identification is not as simple as may initially appear, as will be shown by the performance of two rule-based baselines in our experiments. One source of difficulty is variations in expressions' usage in text. Although MWEs generally show less variation than single words, they show enough that it cannot be ignored. In a study on V+NP idioms, Riehemann (2001) found that the idioms' canonical form accounted for 75% of their appearances in a corpus. Additionally, expressions differ considerably in the types of variations they allow, which include passivization, nominalization and addition of modifying words (Moon, 1998).

A second source of difficulty is that expressions consisting of very frequent words will often co-occur in sentences in a non-MWE usage and in similar but distinct expressions.

MWE identification can be modeled as a two step process. Given a sentence and a known expression, step (1) is to decide if the sentence contains a potential use of the expression. This is a relatively simple step based on the appearance in the sentence of the words comprising the MWE. Step (2) is to decide if the potential use is indeed non-compositional. Consider the following sentences with regard to the expression *hit the road*, meaning 'to leave on a journey':

(a) 'At the time, *the road* was long and difficult with few travelers daring to take it.'

(b) 'The headlights of the taxi-van behind us

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 468–477,
Singapore, 6-7 August 2009.

flashed as it *hit* bumps in *the road*.'

(c) 'The bullets were *hitting the road* and I could see them coming towards me a lot faster than I was able to reverse.'

(d) 'Lorry trailers which would have been *hitting the road* tomorrow now stand idle.'

Sentence (a) does not contain a potential use of the expression due to the missing component 'hit'. Each of (b)-(d) does contain a potential use of the expression. In (b) all of the expression components are present, but they do not form an expression. In (c), the words form an expression, but with a compositional (literal) meaning. Only (d) contains a non-compositional use of *hit the road*. The task we address in this paper is to identify whether or not we are in case (d), for a given expression in a given sentence.

To date, most work in MWE identification has focused on manually encoding rules that identify expressions in text. The encodings, usually consisting of regular expressions and syntactic structures, are intended to contain all the necessary information for processing the MWE in text. Being manual, this is time-consuming work and requires expert knowledge of individual expressions. In terms of the above model, such encodings handle both MWE identification steps.

A second approach is to use machine learning methods to learn an expression's behavior from a corpus. Studies taking this approach have focused on distinguishing between compositional and non-compositional uses of an expression (cases (c) and (d) above). As will be detailed in Section 2, existing methods are tailored to an expression's type, and experiment with a single MWE pattern. In addition, the training and test sets they used did not contain non-expression uses as in case (b), which can be quite common in practice.

Our approach is more general. Given a set of sentences with potential MWE uses, we use sentence surface features to create a Support Vector Machine (SVM) classifier for each expression. The classifier is binary and differentiates between non-compositional uses of the expression ((d) above) on the one hand, and compositional and non-expression uses ((b) and (c)) on the other. The experiments and results presented below focus on verbal MWEs, since verbal MWEs are quite common in language use and have also been investigated in related MWE research (e.g., (Cook et al., 2007)). However, the developed features are not specific to a particular type of expression.

The supervised method is compared with two simple rule-based baselines in order to test whether a simple approach is sufficient. In addition, the use of surface features is compared with the use of syntactic features (based on dependency parse trees of the sentences). Averaged over expressions in an independent test set, the supervised classifiers outperform the rule-based baselines, with F-scores of 94.86% (surface features) and 87.77% (syntactic features), compared with 80.70% for the best baseline.

Section 2 reviews previous work. Section 3 discusses the features used for the supervised classifier. Section 4 explains the experimental setting. The results and a discussion are given in sections 5 and 6.

2 Previous Work

2.1 MWE Lexical Encoding

The approach to handling MWEs in early systems was to employ a list of expressions, each with a quasi regular expression that encodes morpho-syntactic variations. One example is Leech et al. (1994) who used this method for automatic part-of-speech tagging for the BNC. Another is a formalism called IDAREX (IDioms And Regular EXpressions) (Breidt et al., 1996).

More recent research emphasizes the integration of MWE lexical entries into existing single word lexicons and grammar systems (Villavicencio et al., 2004; Alegria et al., 2004). There is also an attempt to take advantage of regularities in morpho-syntactic properties across MWE groups, which allows encoding the behavior of the group instead of individual expressions (Villavicencio et al., 2004; Grégoire, 2007). Fellbaum (1998) discusses some difficulties in representing idioms, which are largely figurative in meaning, in WordNet. More recent work (Fellbaum et al., 2006) focuses on German VP idioms.

As already mentioned, one issue with lexical encoding is that it is done manually, making lexicons difficult to create, maintain and extend. The use of regularities among different types of MWEs is one way of reducing the amount of work required. A second issue is that implementations tend to ignore the likelihood and even the possibility of compositional and other interpretations of expressions in text, which can

be common for some expressions. For example, in an MWE identification study, Hashimoto et al. (2006) built an identification system using hand crafted rules for some 100 Japanese idioms. The results showed near perfect performance on expressions without compositional/non-compositional ambiguity but significantly poorer performance on expressions with ambiguity.

2.2 MWE Identification by ML

Katz and Giesbrecht (2006) used a supervised learning method to distinguish between compositional and non-compositional uses of an expression (in German text) by using contextual information in the form of Latent Semantic Analysis (LSA) vectors. LSA vectors of compositional and non-compositional meaning were built from a training set of example sentences and then a nearest neighbor algorithm was applied on the LSA vector of one tested MWE. The technique was tested more thoroughly in Cook et al. (2007).

Cook et al. (2007) devised two unsupervised methods to distinguish between compositional (literal) and non-compositional (idiomatic) tokens of verb-object expressions. The first method is based on an expression's canonical form. In a previous study (Fazly and Stevenson, 2006), the authors came up with a dozen possible syntactic forms for verb-object pairs (based on passivization, determiner, and object pluralization) and used a corpus-based statistical measure to determine the canonical form(s). The method classifies new tokens as idiomatic if they use a canonical form, and literal otherwise.

The second method uses context as well as form. Co-occurrence vectors representing the idiomatic and literal meaning of each expression were computed based on corpus data. Idiomatic-meaning vectors were based on examples matching the expressions' canonical form. Literal meaning vectors were based on examples that did not match the canonical form. New tokens were classified as literal/idiomatic based on their (co-occurrence) vector's cosine similarity to the idiomatic and literal vectors.

(Sporleder and Li, 2009) also attempted to distinguish compositional from non-compositional uses of expressions in text. Their assumption was that if an expression is used literally, but not idiomatically, its component words will be related semantically to several words in the surrounding discourse. For example, when the expression 'play with fire' is used literally, words such as 'smoke, 'burn', 'fire department', and 'alarm' tend to also be used nearby; when it is used idiomatically, they aren't (indeed, other words, e.g., 'danger' or 'risk' appear nearby but they are not close semantically to 'play' or to 'fire'). This property was used to distinguish literal and non-literal instances by measuring the semantic relatedness of an expression's component words to nearby words in the text. If one or more of the expression's components were sufficiently related to enough nearby words, forming a 'lexical chain', the usage was classified as literal. Otherwise it was idiomatic. Two classifiers based on lexical chains were devised. These were compared with a supervised method that trains a classifier for each expression based on surrounding context. The results showed that the supervised classifier method did much better (90% F-score on literal uses) than the lexical chain classifier methods (60% F-score).

In the above studies the focus is on the compositional/non-compositional expression distinction. The sentence data used contains examples of either one or the other. In (Sporleder and Li, 2009) the experimental data included only sentences in which the expressions were in canonical form (allowing for verb inflection). In (Cook et al., 2007) a syntactic parser was used to collect sentences containing the MWEs in the active and passive voice using heuristics. Thus, examples such as the following (from the BNC) would not be included in their sample:

1. *take a chance*: 'While he still had **a chance** of being near Maisie, he would **take** it'.
2. *face the consequences*: '...she did not have to **face**, it appears, **the** possible serious or even fatal **consequences** of her decision'.
3. *make a distinction*: 'Logically, **the distinction** between the two aspects of the theory can and should be **made**'.
4. *break the ice*: '**The ice**, if not **broken**, was beginning to soften a little'.
5. *settle a score*: 'Morrissey had another **score to settle**'.

This means that their experiments have not included all types of sentences that might be encountered in practice when attempting MWE identifi-

cation. Specifically, they would miss many examples in which the MWE words are present but are not used as an expression (case (b) in Section 1). Moreover, their heuristics are tailored to the Verb-Direct Object MWE type. Different heuristics would need to be employed for different MWE types.

In our approach there is no pre-processing stage requiring type-specific knowledge. Specifically, the above examples are used as training sentences in our experiments.

2.3 MWE Extraction

There exists an extensive body of research on MWE extraction (see Wermter and Hahn (2004) for a review), where the only input is a corpus, and the output is a list of MWEs found in it. Most methods collect MWE candidates from the corpus, score them according to some association measure between their components, and accept candidates with scores passing some threshold. The focus of research has been on developing association measures, including statistical, information-theoretic and linguistically motivated measures (e.g., Justeson and Katz (1995), Wermter and Hahn (2006), and Deane (2005)).

3 MWE Identification Method

Our method decides if a potential use of a known expression in a given sentence is non-compositional. The input to the method, for each MWE, is a labeled training set of sentences containing one or more potentially non-compositional uses of the MWE. The output, for each MWE, is a binary classifier, trained on those sentences. Thus, we target step (2) of MWE identification, which is the difficult one.

The learning algorithm used is Support Vector Machine (SVM), which outputs a binary classifier, using Sequential Minimal Optimization (Platt, 1998)[1] in the Weka toolkit[2] (Witten and Frank, 2000).

For training, sentences are converted into feature vectors. Features depend on the assignment of the lexical components of the expression to specific tokens in the sentence. In some cases, there are several tokens in the sentence that match a single component in the expression, and this leads to multiple (potential) assignments. So in the general case a sentence is converted to a set of feature vectors, each corresponding to a single assignment of the MWE's lexical components to sentence tokens.

Training sentences are labeled positive if they contain a non-compositional use of the expression and negative if they do not (i.e., literal and other uses). If the sentence is positive, at least one of the assignments is the true assignment (there may be more than one, e.g., when an expression is used twice in the same sentence). The vector matching the true assignment is labeled positive. The others are labeled negative. If the sentence is negative, all of the vectors are labeled negative.

As mentioned, the output of the method is a distinct binary classifier for each MWE. Although having a single classifier for all expressions would seem advantageous, the wide variation exhibited by MWEs (e.g., for some the passive is common, for other not at all) precludes this option and requires having a separate classifier for each expression.

3.1 Features

Surface features include order and distance, part-of-speech and inflection of an expression's words in a sentence.

Use of surface features is intuitive and relatively cheap. In addition, many studies have shown the importance of order and distance in MWE extraction in English (two recent examples are (Dias, 2003; Deane, 2005)). Thus, we develop a supervised classifier based on surface features.

Many of the surface features make use of an expression's Canonical Form (CF), thus the learning algorithm assumes that it is given such a form. Formally defining the CF is difficult. Indeed, some researchers have concluded that some expressions do not have a CF (Moon, 1998). For our purposes, CF can be informally defined as the most frequent form in which the expression appears. In practice, an approximation of this definition, explained in Section 4, is used.

3.1.1 Surface Features

1. Word Distance: The number of words between the leftmost and rightmost MWE tokens in the sentence.

2. Ordered Gap List: A list of gaps, measured in number of words, between each pair of the

[1] Using the PUK kernel (The Pearson VII function-based Universal Kernel), with parameters omega=1.0 and sigma=1.0.

[2] Weka version 3.5.6; www.cs.waikato.ac.nz/ ml/ weka/

expression's tokens in their canonical form order. For example, if the token locations (in canonical form order) are 10, 7 and 3, the ordered gap list would be $(10 \leftrightarrow 7 = 2, 10 \leftrightarrow 3 = 6, 7 \leftrightarrow 3 = 3)$.

3. Word Order: A boolean value indicating whether the expression's word order in the sentence matches the canonical form word order.

4. Word Order Permutation: The permutation of word order relative to the canonical form. For example, the permutation (1,0,2) indicates that component words 1 and 0 have switched order in the sentence.

5. Inflection Ratio: The fraction of words in the expression that have undergone inflection relative to the canonical form.

6. Lexical Values: A list of the tokens in the sentence matching the expression's component words, ordered according to canonical form. For example, if the expression is 'make a distinction', a possible lexical values list is (made,no,distinction) in the sentence 'No possible distinction can be made between the two'.

7. POS Pattern: A boolean value indicating whether the expression's use in the sentence has the same part-of-speech pattern as the canonical form.

Two combinations of surface features are used in the experiments below. The first, named R1, uses all of the above features. The second, R2, uses only Word Distance, Ordered Gap List and Word Order Permutation. Using R2 the learner has only word order and distance information from which to create a classifier.

3.1.2 Syntactic Features

An expression's words may appear unrelated in a sentence, because of distance, order, part-of-speech and other surface variations. However, the words will still be closely related syntactically. Syntactic analysis of the sentence in the form of a dependency parse tree directly gives the syntactic relationships between the expression's components. Thus, we also develop a classifier based on syntactic features.

Dependency Parsing. A dependency parse tree is a directed acyclic graph in which the nodes represent tokens in the sentence and the edges represent syntactic dependencies between the words (e.g., direct-object, prepositional-object, noun-subject etc.). The Stanford Parser[3] (Marneffe et al., 2006) was used.

Minimal Sub-Tree. To compute a syntactic feature, the dependency tree is computed and then the minimal sub-tree containing the expression's tokens is extracted.

The features are:

1. Sub-Tree Distance Sum: The number of edges in the minimal sub-tree. A large number of edges suggests a weaker dependency.

2. Sub-Tree Distance List: A list of the distances of the MWE component nodes from the root of their sub-tree.

3. Descendant Relations List: A list of descendant relations between each pair of MWE component nodes.
   A descendant relation between two nodes exists if there is a directed path from one node (the ancestor) to the other (the descendant). Descendant relations are either direct (parent-child) or indirect. The list consists of the levels of descendant relations between the MWE component nodes, which can be none, indirect or direct.

4. Descendant Direction List: A list of the directions of the descendant relations between each pair of MWE component nodes.
   If there are descendant relations between a pair of nodes, the direction of the dependency, indicating which is the modifying and which the modified node, is important.

5. Sibling Relations List: A list of sibling relations between each pair of MWE component nodes.
   Two nodes are first degree siblings if they share the same parent (which usually means they modify the same word). Two nodes are second degree siblings if they share a common ancestor no more than two edges away, and so on. The list consists of the level of sibling relations for each pair of component

[3] http://www-nlp.stanford.edu/software/lex-parser.shtml

nodes, which can be first, second and third degree.

6. Descendant Type List: A list of the dependency types (e.g., subject, direct object etc.) between each pair of component nodes. If the component nodes are not direct descendants their dependency type is null.

7. Sibling Type List: A list of pairs of dependency types corresponding to the dependencies between a pair of component nodes and their common parent. If the component nodes are not first degree siblings, the type is null.

In the experiments reported below, the classifier using only the syntactic features is denoted by S, and the one using all surface and all syntactic features is denoted by C. We have experimented with additional feature combinations, with no improvement in results.

4 Experimental Method

Canonical form. As described, an expression's canonical form (CF) is used in many of the learning algorithm's features. The CF is taken from Collins COBUILD Advanced Learner's English Dictionary (2003) which is also used as our source for MWEs. COBUILD is an English-English dictionary based on the Bank of English (BOE) corpus (over 520 million words) with approximately 34,000 entries.

Traditional single-word dictionaries are a good source for expressions because they usually list, as part of single-word entries, expressions in which the word is a component. The CF is not explicitly given in COBUILD, so an approximation is the form which appears in the expression's definition. This is a reasonable approximation since the COBUILD authors claim to have selected *typical* uses of the expressions in their definitions.

Each CF also has a matching part-of-speech (POS) pattern, which is a list of the parts-of-speech of the components in the CF. For example, 'walking on air' has the pattern $(Verb, Preposition, Noun)$. COBUILD does not include part-of-speech information for expressions so this information was determined using the British National Corpus (BNC) (BNC, 2001), a (mostly) automatically POS tagged corpus (using the CLAWS tagger). For each MWE, the POS patterns of all instances of the CF in the corpus were counted. The most frequent pattern is the expression's POS pattern.

The expressions. A set of 17 verbal MWEs, the development set, was used for development of the surface and syntactic features described above. All of the development set MWEs had the POS pattern $(Verb, Determiner, Noun)$. Another set of 24 verbal MWEs, the training/test set[4], was then used to test the method. Because the method is not specific to the $(Verb, Determiner, Noun)$ pattern, new POS patterns are included in the training/test set. The training/test set consists of 8 MWEs of the POS pattern $(Verb, Determiner, Noun)$, 7 $(Verb, Preposition, Noun)$ MWEs and and 9 $(Verb, Noun, Preposition)$ MWEs. The list of MWEs was selected randomly from the corresponding POS pattern types. MWEs with a positive or negative percentage of under 5% in their data set were discarded[5]. The MWEs, in their canonical form, are:
Development set:
$(Verb, Determiner, Noun)$ [17]: *break the ice, calls the shots, catch a cold, clear the air, face the consequences, fits the bill, hit the road, make a face, make a distinction, makes an impression, raise the alarm, set an example, sound the alarm, stay the course, take a chance, take the initiative, tie the knot*.
Training/test set:
$(Verb, Determiner, Noun)$ [8]: *changes the subject, get a grip, get the picture, lead the way, makes the grade, sets the scene, take a seat, take the plunge*;
$(Verb, Preposition, Noun)$ [7]: *fall into place, goes to extremes, brought to justice, take to heart, gets on nerves, keep up appearances, comes to light*;
$(Verb, Noun, Preposition)$ [9]: *take aim at, make allowances for, takes advantage of, keep hands off, lay claim to, take care of, make contact with, gives rise to, wash hands of.*

The sentences. As mentioned, the first step of MWE identification is to identify if the sentence contains a potential non-compositional use of the expression. In order to test our method, which targets step (2), a set of such sentences (for each expression) was collected from the BNC corpus and

[4] Using 10-fold cross validation.

[5] Initially there were 20 MWEs in the development set and 30 (10 per group) in the training/test set.

then labeled for use as training/test sentences[6].

The collection method was intended to allow a wide range of variations in expression use. In practice, for each expression sentences containing all of the expression's CF components, in any of their inflections, were collected, but excluding common auxiliary words. So for example, when targeting the MWE 'make an impression' we allowed inflections of 'make' and 'impression' and did not require 'an', to allow for variations such as 'make no impression' and 'make some impression'. For some expressions, sentences were limited to those with a distance of up to 8 words between each expression component. Very long sentences (above 80 words) were discarded. The final set of sentences was then randomly selected.

Given this method, training/test sentences allow non-lexical variations: inflection, word order, part-of-speech, syntactic structure and other non-syntactic transformations. Lexical variations which involve a change in one of the expression's components are not allowed, except for common auxiliary words.

For the development set an average of 97 (40-137) sentences were collected per MWE, giving a total of 1663 sentences, with a micro average of 49% positive labels. For the training/test set there were 139 (73-150) sentences per MWE on average, totaling 3350, with a 40% average positive ratio.

The sentences were manually labeled as positive if they contained a non-compositional use of the MWE and negative if they contained a compositional or non-expression usage. Judgment was based on a single sentence, without wider context.

Baseline methods. Two baseline methods are used to test the intuitive notion that simple rule-based methods are sufficient for MWE identification as well as for comparison with the supervised learning methods.

The first method, CanonicalForm (CF), accepts a sentence use as a non-compositional MWE use if and only if the MWE is in canonical form (there are no intervening words between the MWE components, their order matches canonical-form order, and there is an inflection in at most one component word).

The second method, DistanceOrder (DO), accepts a sentence use if and only if the number of words between the leftmost and rightmost MWE components is less than or equal to 2 (not counting the middle MWE component), and if the order matches the canonical form order.

[6]The PyLucene software package, http://pylucene. osafoundation. org/, was used for building an index to the BNC and for searching.

	CF	DO	R1	R2	S	C
Verb-Det-Noun: All (17)						
A	73.53	82.27	89.48	**90.83**	88.58	87.02
P	**97.09**	89.29	82.71	87.18	83.89	78.54
R	58.81	76.83	92.29	90.35	92.97	**97.19**
F	67.39	79.68	86.92	**88.56**	87.78	86.00
Verb-Det-Noun: Best (8)						
A	84.51	91.56	95.33	**95.48**	92.52	93.27
P	**95.90**	85.70	92.50	95.63	91.12	87.63
R	73.50	89.80	97.25	95.25	95.83	**98.50**
F	78.63	86.29	94.70	**95.36**	93.44	92.25

Table 1: Development set: Average performance over all MWEs and best 8. Supervised classifiers outperform baselines. **A**: Accuracy; **P**: Positive Precision; **R**: Positive Recall; **F**: F-Score.

5 Results

The baseline methods (CF and DO) and the supervised methods (R1,R2,S,C) were run on the development and training/test sets. For the supervised methods, for each MWE we used 10-fold cross-validation[7].

Tables 1 and 2 summarize the results for the development and test sets, respectively. For the development set, average results over all 17 MWEs and over the best 8 MWEs (on R1), a group size comparable to the test set, are shown. For the test set, results over all 24 MWEs and the three MWE types tested are shown.

The tables show average overall accuracy and average precision, recall and F-score on positive instances, where the averages are taken over the results of the individual MWEs (i.e., micro-averaged).

Baselines. Baseline accuracy, (for DO) 82.27% on the development set and 87.2% on the test set (over all groups), is probably insufficient for many NLP applications.

The baselines perform similarly in terms of average accuracy. CF does this with very high precision and low recall, while for DO recall improves at the expense of precision. Looking at individual MWEs reveals that for expressions which allow more variation in terms of intervening words

[7]I.e., we ran 10 experiments where in each experiment we divided the corresponding annotated sentence sets into 90% training sentences and 10% test sentences, and the results reported are the average of the 10 experiments.

	CF	DO	R1	R2	S	C
All (24)						
A	86.16	87.15	**93.50**	91.61	89.73	91.50
P	**94.16**	80.38	93.08	93.16	89.86	89.26
R	68.86	86.88	93.00	89.74	88.94	**93.33**
F	75.53	80.70	**94.86**	93.09	87.77	92.80
Verb-Det-Noun (8)						
A	89.08	89.08	**93.83**	93.65	90.07	91.33
P	**95.44**	84.13	92.88	94.00	91.04	89.25
R	73.30	88.53	97.50	95.50	91.57	**97.63**
F	80.97	84.91	**95.09**	94.71	91.21	93.08
Verb-Prep-Noun (7)						
A	85.53	91.15	**93.64**	92.62	88.75	92.10
P	97.13	81.40	96.81	**97.20**	92.48	94.33
R	64.36	**92.67**	84.73	82.79	82.71	85.00
F	74.08	86.03	**97.81**	96.87	83.13	96.65
Verb-Noun-Prep (9)						
A	84.06	82.32	**93.11**	88.99	90.18	91.18
P	90.72	76.26	**90.78**	89.73	86.78	85.89
R	68.41	80.90	95.44	90.03	91.44	**96.00**
F	71.82	72.82	**92.69**	89.14	88.33	89.99

Table 2: Test set: Average performance over all MWEs and by group. The best supervised classifier outperforms baselines in all groups. **A**: Accuracy; **P**: Precision; **R**: Recall; **F**: F-Score.

and lexical change, DO outperforms CF. To name a few, *make an impression, raise the alarm, take a chance* and *make allowances for*. For example, for *take a chance* intervening words are quite common, as in: 'I'm taking a real chance on you.', or a change in determiner as in: 'I preferred to take my chances'. Indeed, CF showed poor precision only for MWEs with a common literal usage. Two such MWEs were present in the development set (*break the ice* and *tie the knot*) and two in the test set (*wash hands of* and *keep hands off*).

Baselines versus supervised classifiers. As shown in the tables, R1 outperforms the best baseline in terms of accuracy in both test and development. Moreover, the supervised classifiers are more stable in their accuracy. For the development set the standard deviation of accuracy scores averages 22.58 for CF and DO, and 6.68 for R1, R2, S, and C. For the test set the baselines average 9.07 (Verb-Det-Noun), 11.11 (Verb-Prep-Noun) and 14.26 (Verb-Noun-Prep), and the supervised methods average 4.97 (Verb-Det-Noun), 7.66 (Verb-Prep-Noun) and 7.97. This stability means that the supervised classifiers are able to perform well on MWEs with different behavior. For example, R1 is able to perform well on expressions where order is strict, as DO does (e.g., *make a face*), while also performing well on those where order varies (e.g., *make a distinction*).

Supervised classifiers. R1 and R2, based on surface features, show similar accuracy values, with R1 doing somewhat better in the Verb-Prep-Noun and Verb-Noun-Prep groups. This is due to the Lexical Values feature, which accounts for a change in preposition. A change in preposition (as in 'wash hands *of* some matter' versus 'wash hands *in* the sink') is more significant than a change in determiner in the Verb-Determiner-Noun group. This improves precision on negative instances, which are rejected more precisely based on the preposition value. Nevertheless, the relatively simple features in R2, essentially order and distance, perform quite well.

The F-score result for R1, 94.86, is an improvement over the F-score result of the supervised classifier used in (Sporleder and Li, 2009), 90.15. Although the sentence data is different (our data includes sentences with non-expression uses) the number of sentences used is similar.

S, based on syntactic features, performs worse than R1/2. It shows better accuracy than the baselines in all but the (Verb-Prep-Noun) group and is also more stable. C, a combination of surface and syntactic features, performs better than S and slightly worse than R1/2.

Why do the syntactic features perform worse than surface features? An analysis of the S classifier errors reveals two important causes. First, there is substantial variation in the dependency tree structures of the non-compositional uses of the expressions as output by the parser. Thus, the syntactic feature classifier was more difficult to learn than the surface feature one, requiring a larger training set. This is not surprising, given that many MWEs exhibit an irregular syntactic behavior that might even seem strange at times. For example, in the sentence fragment "and then he came to.", 'came to' is an MWE. A parser might find it difficult to parse the sentence correctly, expecting a noun phrase to follow the 'to'.

Second, as described above, the syntactic features consist of general syntactic relations extracted from the parse tree and not type-specific knowledge. As a result, literal or non-expression uses of the MWE's components, which have a close syntactic relation in a given sentence, appear as non-compositional uses of the expression to the classifier.

6 Discussion

This study has addressed MWE identification: deciding if a potential use of an expression is a non-compositional one. Despite its importance in basic NLP tasks, the problem has been largely overlooked in NLP research, probably due to it presumed simplicity. However, as we have shown, simple methods for MWE identification, such as our baselines, do not perform consistently well across MWEs. This study serves to highlight this point and the need for more sophisticated methods for MWE identification.

We have shown that using a supervised learning method employing surface sentence features based on canonical form, it is possible to improve performance significantly. Unlike previous research, our method is not tailored to specific MWE types, and we did not ignore non-expression uses in our experiments.

Future research should experiment with non-verbal MWEs, since our features are not specific to verbal MWE types. Another direction is a more sophisticated corpus sampling algorithm. The current work ignored MWEs which had an unbalanced training set (usually too few positives). Methods for gathering enough positive instances of such MWEs will be useful for testing the methods proposed here, as well as for general MWE research.

References

Iñki Alegria, Olatz Ansa, Xabier Artola, Nerea Ezeiza, Koldo Gojenola and Ruben Urizar. 2004. Representation and treatment of multiword expressions in Basque. *ACL '04 Workshop on Multiword Expressions*.

The British National Corpus. 2001. *The British National Corpus, version 2 (BNC World)*. Distributed by Oxford University Computing Services on behalf of the BNC Consortium. URL: http://www.natcorp.ox.ac.uk/

Elisabeth Breidt, Frederique Segond, and Giuseppen Valetto. 1996. Local grammars for the description of multi-word lexemes and their automatic recognition in texts. *COMPLEX '96*. Budapest.

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2007. Pulling their weight: Exploiting syntactic forms for the automatic identification of idiomatic expressions in context. *ACL '07 Workshop on A Broader Perspective on Multiword Expressions*.

Collins COBUILD. 2003. *Collins COBUILD Advanced Learner's English Dictionary*. HarperCollins Publishers, 4th edition.

Paul Deane. 2005. A nonparametric method for extraction of candidate phrasal terms. *ACL '05*.

Gael Dias. 2003. Multiword unit hybrid extraction. *ACL '03 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*.

Afsaneh Fazly and Suzanne Stevenson. 2006. Automatically constructing a lexicon of verb phrase idiomatic combinations. *EACL '06*.

Christiane Fellbaum, Alexander Geyken, Axel Herold, Fabian Koerner, and Gerald Neumann. 2006. Corpus-based studies of German idioms and light verbs. *International Journal of Lexicography*, 19(4):349–361.

Christiane Fellbaum. 1998. Towards a representation of idioms in WordNet. *COLING-ACL '98 Workshop on the Use of WordNet in Natural Language Processing Systems*.

Nicole Grégoire. 2007. Design and implementation of a lexicon of Dutch multiword expressions. *ACL '07 Workshop on A Broader Perspective on Multiword Expressions*.

Chikara Hashimoto, Satoshi Sato, and Takehito Utsuro. 2006. Japanese idiom recognition: Drawing a line between literal and idiomatic meanings. *COLING-ACL '06, Poster Sessions*.

John S. Justeson and Slava M. Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1:9–27.

Graham Katz and Eugenie Giesbrecht. 2006. Automatic identification of non-compositional multiword expressions using latent semantic analysis. *COLING-ACL '06 Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties.*.

Geoffrey Leech, Roger Garside and Michael Bryant. 1994. CLAWS4: The tagging of the British National Corpus. *COLING '94*.

Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. *LREC '06*.

Rosamund Moon. 1998. *Fixed Expressions and Idioms in English*. Oxford: Clarendon Press.

John Platt. 1998. Machines using sequential minimal optimization. In *In B. Schoelkopf and C. Burges and A. Smola, editors, Advances in Kernel Methods – Support Vector Learning*.

Susanne Z. Riehemann. 2001. *A Constructional Approach to Idioms and Word Formation*. Ph.D. Thesis. Stanford.

Caroline Sporleder and Linlin Li. 2009. Unsupervised recognition of literal and non-literal use of idiomatic expressions. *EACL '09.*

Aline Villavicencio, Ann Copestake, Benjamin Waldron, and Fabre Lambeau. 2004. Lexical encoding of MWE. *ACL '04 Workshop on Multiword Expressions*.

Joachim Wermter and Udo Hahn. 2004. Collocation extraction based on modifiability statistics. *COLING '04.*

Joachim Wermter and Udo Hahn. 2006. You can't beat frequency (unless you use linguistic knowledge) – a qualitative evaluation of association measures for collocation and term extraction. *COLING-ACL '06*.

Ian H. Witten amd Eibe Frank. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

Acquiring Translation Equivalences of Multiword Expressions by Normalized Correlation Frequencies

Ming-Hong Bai[1,2] **Jia-Ming You**[1] **Keh-Jiann Chen**[1] **Jason S. Chang**[2]
1 Institute of Information Science, Academia Sinica, Taiwan
2 Department of Computer Science, National Tsing-Hua University, Taiwan
mhbai@sinica.edu.tw, swimming@hp.iis.sinica.edu.tw,
kchen@iis.sinica.edu.tw, jschang@cs.nthu.edu.tw

Abstract

In this paper, we present an algorithm for extracting translations of any given multiword expression from parallel corpora. Given a multiword expression to be translated, the method involves extracting a short list of target candidate words from parallel corpora based on scores of normalized frequency, generating possible translations and filtering out common subsequences, and selecting the top-n possible translations using the Dice coefficient. Experiments show that our approach outperforms the word alignment-based and other naive association-based methods. We also demonstrate that adopting the extracted translations can significantly improve the performance of the Moses machine translation system.

1 Introduction

Translation of multiword expressions (MWEs), such as compound words, phrases, collocations and idioms, is important for many NLP tasks, including the techniques are helpful for dictionary compilation, cross language information retrieval, second language learning, and machine translation. (Smadja et al., 1996; Gao et al., 2002; Wu and Zhou, 2003). However, extracting exact translations of MWEs is still an open problem, possibly because the senses of many MWEs are not compositional (Yamamoto and Matsumoto, 2000), i.e., their translations are not compositions of the translations of individual words. For example, the Chinese idiom 坐視不理 should be translated as "*turn a blind eye,*" which has no direct relation with respect to the translation of each constituent (i.e., *"to sit", "to see" and "to ignore"*) at the word level.

Previous SMT systems (e.g., Brown et al., 1993) used a word-based translation model which assumes that a sentence can be translated into other languages by translating each word into one or more words in the target language. Since many concepts are expressed by idiomatic multiword expressions instead of single words, and different languages may realize the same concept using different numbers of words (Ma et al., 2007; Wu, 1997), word alignment based methods, which are highly dependent on the probability information at the lexical level, are not well suited for this type of translation.

To address the above problem, some methods have been proposed for extending word alignments to phrase alignments. For example, Och et al. (1999) proposed the so-called *grow-diag-final* heuristic method for extending word alignments to phrase alignments. The method is widely used and has achieved good results for phrase-based statistical machine translation. (Och et al., 1999; Koehn et al., 2003; Liang et al., 2006). Instead of using heuristic rules, Ma et al. (2008) showed that syntactic information, e.g., phrase or dependency structures, is useful in extending the word-level alignment. However, the above methods still depend on word-based alignment models, so they are not well suited to extracting the translation equivalences of semantically opaque MWEs due to the lack of word level relations between the translational correspondences. Moreover, the aligned phrases are not precise enough to be used in many NLP applications like dictionary compilation, which require high quality translations.

Association-based methods, e.g., the *Dice coefficient*, are widely used to extract translations of MWEs. (Kupiec, 1993; Smadja et al., 1996; Kitamura and Matsumoto, 1996; Yamamoto and Matsumoto, 2000; Melamed, 2001). The advantage of such methods is that association relations are established at the phrase level instead of the lexical level, so they have the potential to resolve the above-mentioned translation problem. However, when applying association-based methods, we have to consider the following complications. The first complication, which we call the *contextual effect*, causes the extracted translation to contain noisy words. For

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 478–486,
Singapore, 6-7 August 2009.

example, translations of the Chinese idiom *兩全其美* (*best of both worlds*) extracted by a naive association-based method may contain noisy collocation words like *difficult*, *try* and *cannot*, which are not part of the translation of the idiom. They are actually translations of its collocation context, such as *難以*(*difficult*), *嘗試*(*try*), and *不能*(*cannot*). This problem arises because naive association methods do not deal with the effect of strongly collocated contexts carefully. If we can incorporate lexical-level information to discount the noisy collocation words, the contextual effect could be resolved.

English (y)	f_y	$f_{x,y}$	Dice(x,y)
quote **out of context**	22	19	0.56
take **out of context**	17	11	0.35
interpret **out of context**	2	2	0.08
out of context	53	32	0.65

Table 1. The Dice coefficient tends to select a common subsequence of translations. (The frequency of *斷章取義*, f_x, is 46.)

The second complication, which we call the *common subsequence problem*, is that the Dice coefficient tends to select the common subsequences of a set of similar translations instead of the full translations. Consider the translations of *斷章取義* (*quote out of context*) shown in the first three rows of Table 1. The Dice coefficient of each translation is smaller than that of the common subsequence "*out of context*" in the last row. If we can tell common subsequence apart from correct translations, the common subsequence problem could be resolved.

In this paper, we propose an improved precision method for extracting MWE translations from parallel corpora. Our method is similar to that of Smadja et al. (1996), except that we incorporate lexical-level information into the association-based method. The algorithm works effectively for various types of MWEs, such as phrases, single words, rigid word sequences (i.e., no gaps) and gapped word sequences. Our experiment results show that the proposed translation extraction method outperforms word alignment-based methods and association-based methods. We also demonstrate that precise translations derived by our method significantly improve the performance of the Moses machine translation system.

The remainder of this paper is organized as follows. Section 2 describes the methodology for extracting translation equivalences of MWEs. Section 3 describes the experiment and presents the results. In Section 4, we consider the application of our results to machine translation. Section 5 contains some concluding remarks.

2 Extracting Translation Equivalences

Our MWE translation extraction method is similar to the two-phase approach proposed by Smadja et al. (1996). The two phases can be briefly described as follows:

Phase 1: Extract candidate words correlated to the given MWE from parallel text.

Phase 2:

1. Generate possible translations for the MWE by combining the candidate words.
2. Select possible translations by the Dice coefficient.

We propose an association function, called the *normalized correlation frequency*, to extract candidate words in the phase 1. This method incorporates lexical-level information with association measure to overcome the *contextual effect*. In phase 2, we also propose a *weighted frequency* function to filter out false common subsequences from possible translations. The filtering step is applied before the translation selecting step of phase 2.

Before describing our extraction method, we define the following important terms used throughout the paper.

Focused corpus (*FC*): This is the corpus created for each targeted MWE. It is a subset of the original parallel corpora, and is comprised of the selected aligned sentence pairs that contain the source MWE and its translations.

Candidate word list (*CW*): A list of extracted candidate words for the translations of the source MWE.

2.1 Selecting Candidate Words

For a source MWE, we try to extract from the *FC* a set of k candidate words *CW* that are highly correlated to the source MWE. We then assume that the target translation is a combination of some words in *CW*. As noted by Smadja et al. (1996), this two-step approach drastically reduces the search space.

However, translations of collocated context words in the source word sequence create noisy candidate words, which might cause incorrect extraction of target translations by naive statistical correlation measures, such as the Dice coef-

ficient used by Smadja et al. (1996). The need to avoid this context effect motivates us to propose a candidate word selection method that uses the normalized correlation frequency as an association measure.

The rationale behind the proposed method is as follows. When counting the word frequency, each word in the target corpus normally contributes a frequency count of one. However, we are only interested in the word counts correlated to a **MWE**. Therefore, intuitively, we define the normalized count of a target word *e* as the translation probability of *e* given the **MWE**.

We explain the concept of normalizing the correlation count in Section 2.1.1 and the computation of the normalized correlation frequency in Section 2.1.2.

2.1.1 Normalizing Correlation Count

We propose an association measure called the *normalized correlation frequency*, which ranks the association strength of target words with the source MWE. For ease of explanation, we use the following notations: let $\mathbf{f}=f_1,f_2,\ldots,f_m$ and $\mathbf{e}=e_1,e_2,\ldots,e_n$ be a pair of parallel Chinese and English sentences; and let $\mathbf{w}=t_1,t_2,\ldots,t_r$ be the Chinese source MWE. Hence, **w** is a subsequence of **f**.

When counting the word frequency, each word in the target corpus normally contributes a frequency count of one. However, since we are interested in the word counts that correlate to **w**, we adopt the concept of the translation model proposed by Brown et al (1993). Each word *e* in a sentence **e** might be generated by some words, denoted as **r**, in the source sentence **f**. If **r** is non-empty the relation between **r** and **w** should fit one of the following cases:

1) All words in **r** belong to **w**, i.e., $\mathbf{r} \subseteq \mathbf{w}$, so we say that *e* is only generated by **w**.
2) No words in **r** belong to **w**, i.e., $\mathbf{r} \subseteq \mathbf{f} - \mathbf{w}$, so we say that *e* is only generated by context words.
3) Some words in **r** belong to **w**, while others are context words.

Intuitively, In Cases 1 and 2, the correlation count of an instance *e* should be 1 and 0 respectively. In Case 3, the normalized count of *e* is the expected frequency generated by **w** divided by the expected frequency generated by **f**. With that in mind, we define the *weighted correlation count*, *wcc*, as follows:

$$wcc(e;\mathbf{e},\mathbf{f},\mathbf{w}) = \frac{\sum_{\forall f_i \in \mathbf{w}} p(e \mid f_i) + \Delta \mid \mathbf{w} \mid}{\sum_{\forall f_j \in \mathbf{f}} p(e \mid f_j) + \Delta \mid \mathbf{f} \mid},$$

where Δ is a very small smoothing factor in case *e* is not generated by any word in **f**. The probability $p(e \mid f)$ is the word translation probability trained by IBM Model 1 on the whole parallel corpus.

The rationale behind the *weighted correlation count*, *wcc*, is that if *e* is part of the translation of **w,** then its association with **w** should be stronger than other words in the context. Hence its *wcc* should be closer to 1. Otherwise, the association is weaker and the *wcc* should be closer to 0.

2.1.2 Normalized Correlation

Once the weighted correlation counts *wcc* is computed for each word in *FC*, we compute the normalized correlation frequency for each word *e* as the total sum of the $wcc(e;\mathbf{e},\mathbf{f},\mathbf{w})$ of all **w** in bilingual sentences (**e**, **f**) in *FC*. The *normalized correlation frequency (ncf)* is defined as follows:

$$ncf(e;\mathbf{w}) = \sum_{i=1}^{n} wcc(e;\mathbf{e}^{(i)},\mathbf{f}^{(i)},\mathbf{w}).$$

We choose the top-n English words ranked by *ncf* as our candidate words and filter out those whose *ncf* is less than a pre-defined threshold. Table 2 shows the candidate words for the Chinese term 斷章取義 (quote/take/interpret out of context) sorted by their *ncf* values. To illustrate the effectiveness *ncf*, we also display candidate words of the term with their Dice values in Tables 3. As shown in the tables, noise words such as *justify, meaning* and *unfair* are ranked lower using *ncf* than using Dice, while correct candidates, such as *out*, *take* and *remark* are ranked higher. We present more experimental results in Section 3.

2.2 Generation and Ranking of Candidate Translations

After determining the candidate words, candidate translations of **w** can be generated by marking the candidate words in each sentence of *FC*. The word sequences marked in each sentence are deemed possible translations. At the same time, the weakly associated function words,

Candidate words *e*	freq	*ncf*(*e*,**w**)
context	54	31.55
out	58	24.58
quote	26	5.84
take	23	4.81
remark	8	1.84
interpret	3	1.38
piecemeal	1	0.98
deliberate	3	0.98

Table 2. Candidate words for the Chinese term *斷章取義* sorted by their global normalized correlation frequencies.

Candidate words *e*	freq	*dice*(*e*,**w**)
context	54	0.0399
quote	26	0.0159
deliberate	3	0.0063
justify	3	0.0034
interpretation	7	0.0032
meaning	3	0.0029
cite	3	0.0025
unfair	4	0.0023

Table 3. Candidate words for the Chinese term *斷章取義* sorted by their Dice coefficient values.

which we fail to select in the candidate word selection stage, should be recovered. The rule is quite simple: if a function word is adjacent to any candidate word, it should be recovered. For example, in the following sentence, the function word *of* would be recovered and added to the marked sequence:

"The financial secretary has been **quoted out** of **context.**
財政司 司長 之 談話 被 **斷章取義.**"

The marked words are shown in boldface.

2.2.1 Generating Possible Translations

Although we have selected a reliable candidate word list, it may still contain some noisy words due to the MWE's collocation context. Consider the following example:

...as **quoted** in the audit report, if **taken out of context...**

In this instance, *quoted* is a false positive; therefore, the marked word sequence **m** "*quoted taken out of context*" is not the correct translation. To avoid such false positives, we include **m** and all its subsequences as possible translations.

quoted taken out of context
quoted taken out of
quoted taken out context
quoted taken of context
quoted out of context
taken out of context
…
quoted out
taken out
quoted
taken
out
context

Table 4. Example subsequences generated of **w** and add them to the candidate translation list.

Table 4 shows the subsequences of **m** in the above example. The generation process is used to increase the coverage of correct translations in the candidate list; otherwise, many correct translations will be lost. However, the process may also trigger the side effect of the common subsequence problem described in Section 1. Since all candidates compete for the best translations by comparing their association strength with **w,** the common subsequences will have an advantage.

2.2.2 Filtering Common Subsequences

To resolve the *common subsequence effect* problem, we evaluate each candidate translation, including its subsequences, by a concept similar to the normalized correlation frequency. As mentioned in Section 1, the Dice coefficient tends to select the common subsequences of some candidates because they have higher frequencies. To avoid this problem, we use the normalized correlation frequency to filter out false common subsequences from the candidate translation list. Here, we also use the weighted correlation count *wcc* to weight the frequency count of a candidate translation. Suppose we have a marked sequence in a sentence, **m,** whose subsequences are generated in the way described in the previous section. If the weighted count of **m** is assigned the score 1, the *weighted count* (*wc*) of a subsequence **t** is then defined as follows:

$$wc(\mathbf{t};\mathbf{e},\mathbf{f},\mathbf{m},\mathbf{w}) = \prod_{\forall e \in \mathbf{m}-\mathbf{t}} (1 - wcc(e;\mathbf{e},\mathbf{f},\mathbf{w})) .$$

The underlying concept of *wc* is that the original marked sequence **m** is supposed to be the most

likely translation of **w** and the weighted count is set to 1. Then, if a subsequence **t** is generated by removing a word *e* from **m**, the weighted count of the subsequence is reduced by multiplying the complement probability of *e* generated by **w**. Note that the weighted correlation count *wcc* is the probability of the word *e* generated by **w**.

After all $wc(\mathbf{t};\mathbf{e},\mathbf{f},\mathbf{m},\mathbf{w})$ in each sentence of the *FC* have been computed, the *weighted frequency* for a sequence **t** can be determined by summing the weighted frequencies over *FC* as follows:

$$wf(\mathbf{t};\mathbf{w}) = \sum_{\forall(\mathbf{e},\mathbf{f})\in FC} wc(\mathbf{t};\mathbf{e},\mathbf{f},\mathbf{m},\mathbf{w}).$$

We compute the *wf* for each candidate translation and then sort the candidate translations by their *wf* values.

Next, we filter out common subsequences based on the following rule: for a sequence **t**, if there is a super-sequence **t'** on the sorted candidate translation list and the *wf* value of **t** is less than that of **t'**, then **t** is assumed be a common subsequence of real translations and removed from the list.

candidate translation list	freq	wf
quote out of context	19	17.55
of context	35	15.45
out of context	32	14.82
quote of context	19	13.32
out	35	11.92
quote	23	11.63
quote out	19	9.42

Table 5. Part of the candidate translation list for the Chinese idiom, *斷章取義*, sorted by the *wf* values.

Table 5 shows an example of the rule's application. The candidate translation list is sorted by the translations' *wf* values. Then, candidates 2-7 are removed because they are subsequences of the first candidate and their *wf* values are smaller than that of the first candidate.

2.3 Selection of Candidate Translations

Having removed the common subsequences of real translations from the candidate translation list of **w**, we can select the best translations by comparing their association strength with **w** for the remaining candidates. The Dice coefficient is a good measure for assessing the association strength and selecting translations from the candidate list. For a candidate translation **t,** the Dice coefficient is defined as follows:

$$Dice(\mathbf{t},\mathbf{w}) = \frac{2p(\mathbf{t},\mathbf{w})}{p(\mathbf{t})+p(\mathbf{w})}.$$

Where $p(\mathbf{t},\mathbf{w})$, $p(\mathbf{t})$, $p(\mathbf{w})$ are probabilities of (**t**,**w**), **t**, **w** derived from the training corpus.

After obtaining the Dice coefficients of the candidate translations, we select the top-n candidate translations as possible translations of **w**.

3 Experiments

In our experiments, we use the Hong Kong Hansard and the Hong Kong News parallel corpora as training data. The training data was preprocessed by Chinese word segmentation to identify words and parsed by Chinese parser to extract MWEs. To evaluate the proposed approach, we randomly extract 309 Chinese MWEs from training data, including dependent word pairs and rigid idioms. We then randomly select 103 of those MWEs as the development set and use the other 206 as the test set. The reference translations of each Chinese MWE are manually extracted from the parallel corpora.

3.1 Evaluation of Word Candidates

To evaluate the method for selecting candidate words, we use the coverage rate, which is defined as follows:

$$coverage = \frac{1}{n}\sum_{\forall \mathbf{w}} \frac{|A_{\mathbf{w}} \cap C_{\mathbf{w}}|}{|A_{\mathbf{w}}|},$$

where *n* is the number of MWEs in the test set, $A_{\mathbf{w}}$ denotes the word set of the reference translations of **w**, and $C_{\mathbf{w}}$ denotes a candidate word list extracted by the system.

Table 6 shows the coverage of our method, NCF, compared with the coverage of the IBM model 1 and the association-based methods MI, Chi-square, and Dice. As we can see, the top-10 candidate words of NCF cover almost 90% of the words in the reference translations. Whereas, the coverage of the association-based methods and IBM model 1 is much lower than 90%. The result implies that the candidate extraction method can extract a more precise candidate set than other methods.

Method	Top10	Top20	Top30
MI	0.514	0.684	0.760
Chi-square	0.638	0.765	0.828
Dice	0.572	0.735	0.803
IBM 1	0.822	0.900	0.948
NCF	0.899	0.962	0.973

Table 6. The coverage rates of the candidate words extracted by the compared methods

Figure 1 shows the curve diagram of the coverage rate of each method. As the figure shows, when the size of the candidate list is increased, the coverage rate of using NCF rises rapidly as *n* increases but levels off after *n*=20. Whereas, the coverage rates of other measures grow much slowly.

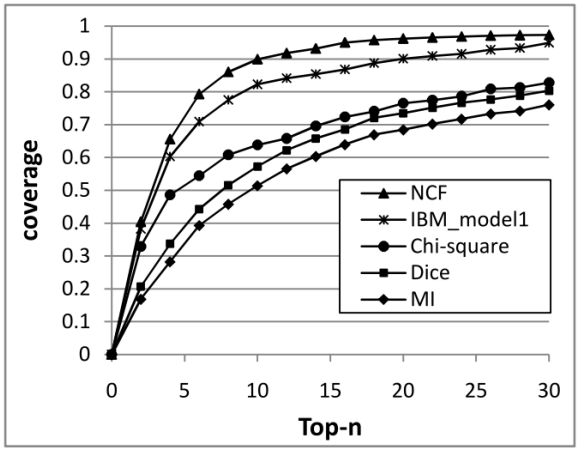


Figure 1. The curve diagram of the coverage of the candidate word list compiled by each method.

From the evaluation of candidate word selection, we find that the *ncf* method, which incorporates lexical-level information into association-based measure, can effectively filter out noisy words and generates a highly reliable list of candidate words for a given MWE.

3.2 Evaluating Extracted Translations

To evaluate the quality of MWE translations extracted automatically, we use the following three criteria:

1) **Translation accuracy:**
   This criterion is used to evaluate the top-n translations of the system. It treats each translation produced as a string and compares the whole string with the given reference translations. If any one of the top-n hypothesis translations is included in the reference translations, it is deemed correct.
2) **WER** (word error rate):
   This criterion compares the top-1 hypothesis translation with the reference translations by computing the edit distance (i.e., the minimum number of substitutions, insertions, and deletions) between the hypothesis translation and the given reference translations.
3) **PER** (position-independent word error rate):
   This criterion ignores the word order and computes the edit distance between the top-1 hypothesis translation and the given reference translations.

We also use the MT task to evaluated our method with other systems. For that, we use the GIZA++ toolkit (Och et al., 2000) to align the Hong Kong Hansard and Hong Kong News parallel corpora. Then, we extract the translations of the given source sequences from the aligned corpus as the baseline. We use the following two methods to extract translations from the aligned results.

1) **Uni-directional alignment**
   We mark all English words that were linked to any constituent of **w** in the parallel Chinese-English aligned corpora. Then, we extract the marked sequences from the corpora and compute the frequency of each sequence. The top-n high frequency sequences are returned as the possible translations of **w**.
2) **Bi-directional alignments**
   We use the *grow-diag-final* heuristic (Och et al., 1999) to combine the Chinese-English and English-Chinese alignments, and then extract the top-n high frequency sequences as described in method 1.

To determine the effect of the *common subsequence filtering method*, FCS, we divide the evaluation of our system into two phases:

1) **NCF+Dice**:
   This system uses the normalized correlation frequency, NCF, to select candidate words as described in Section 2.1. It then extracts candidate translations (described in Section 2.2), but FCS is not used.
2) **NCF+FCS+Dice**:
   This is similar to system 1, but it uses FCS to filter out common subsequences (described in subsection 2.2.2).

Method	WER(%)	PER(%)
Uni-directional	4.84	4.02
Bi-directional	5.84	5.12
NCF+Dice	3.55	3.24
NCF+FCS+Dice	**2.45**	**2.23**

Table 7. Translation error rates of the systems.

Method	Top1	Top2	Top3
Uni-directional	67.5	79.6	83.0
Bi-directional	65.5	77.7	81.1
NCF+Dice	72.8	85.9	88.3
NCF+FCS+Dice	**78.2**	**89.3**	**91.7**

Table 8. Translation accuracy rates of the systems. (%)

Table 7 shows the word error rates for the above systems. As shown in the first and second rows, the translations extracted from *uni-directional* alignments are better than those extracted from *bi-directional* alignments. This means that the *grow-diag-final* heuristic reduces the accuracy rate when extracting MWE translations.

The results in the third row show that the NCF+Dice system outperforms the methods based on GIZA++. In other words, the NCF method can effectively resolve the difficulties of extracting MWE translations discussed in Section 1.

In addition, the fourth row shows that the NCF+FCS+Dice system also outperforms the NCF+Dice system. Thus, the FCS method can resolve the common subsequence problem effectively.

Table 8 shows the translation accuracy rates of each system. The NCF+FCS+Dice system achieves the best translation accuracy. Moreover, it significantly improves the performance of finding MWE translation equivalences.

4 Applying MWE Translations to MT

To demonstrate the usefulness of extracted MWE translations to existing statistical machine translation systems, we use the XML markup scheme provided by the Moses decoder, which allows the specification of translations for parts of a sentence. The procedure for this experiment consists of three steps: (1) the extracted MWE translations are added to the test set with the XML markup scheme, (2) after which the data is input to the Moses decoder to complete the translation task, (3) the results are evaluated using the BLEU metric (Papineni et al., 2002).

	Moses	MWE +Moses
NIST06-sub	23.12	23.49
NIST06	21.57	21.79

Table 9. BLEU scores of the translation results.

4.1 Experimental Settings

To train a translation model for Moses, we use the Hong Kong Hansard and the Hong Kong News parallel corpora as training data (2,222,570 sentence pairs). We also use the same parallel corpora to extract translations of MWEs. The NIST 2008 evaluation data (1,357 sentences, 4 references) is used as development set and NIST 2006 evaluation data (1,664 sentences, 4 references) is used as test set.

4.2 Selection of MWEs

Due to the limitation of the XML markup scheme, we only consider two types of MWEs: continuous bigrams and idioms. Since the goal of this experiment is not focus on extraction of MWEs, simple methods are applied to extract MWEs from the training data: (1) we collect all continuous bigrams from Chinese sentences in the training data and then simply filter out the bigrams by mutual information (MI) with a threshold[1], (2) we also extract all idioms from Chinese sentences of the training data by collecting all 4-syllables words from the training data and filtering out obvious non-idioms, such as determinative-measure words and temporal words by their part-of-speeches, because most Chinese idioms are 4-syllables words.

In total, 33,767 Chinese bigram types and 20,997 Chinese idiom types were extracted from training data; and the top-5 translations of each MWE were extracted by the method described in Section 2. Meanwhile 1,171 Chinese MWEs were added to the translations in the test set. The Chinese words covered by the MWEs in test data set were 2,081 (5.3%).

4.3 Extra Information

When adding the translations to the test data, two extra types of information are required by the Moses decoder. The first type comprises the function words between the translation and its context. For example, if *經貿 合作/economic cooperation* is added to the test data, possible

[1] We set the threshold at 5.

source sentence	... 進入 <MWE>五光十色</MWE> 的社會 ...
Moses	... entered **blinded by the colourful** community ...
MWE+Moses	... entered **the colourful** community ...
reference	... entered **the colorful** society ...
source sentence	... 不希望看到 <MWE>進一步 升級</MWE> 危機 ...
Moses	... do not want to see **an escalation** of crisis ...
MWE+Moses	... do not want to see **a further escalation** of crisis ...
reference	... don 't want to see **the further escalation** of the crisis ...
source sentence	... 廣大 人民 的 <MWE>根本 利益</MWE> ...
Moses	... the people 's **interests** ...
MWE+Moses	... the people of the **fundamental interests** ...
reference	... the **fundamental interests** of the masses ...

Table 10. Examples of improved translation quality with the MWE translation equivalences.

function words, such as '*in*' or '*with*', should be provided for the translation. Because the Moses decoder does not generate function words that are context dependent, it treats a function word as a part of the translation. Therefore, we collect possible function words for each translation from the corpora when the conditional probability is larger than a threshold[2].

The second type of information is the *phrase translation probability* and *lexical weighting*. Computing the phrase translation probability is trivial in the training corpora, but lexical weighting (Koehn et al., 2003) needs lexical-level alignment. For convenience, we assume that each word in an MWE links to each word in the translations. Under this assumption, the lexical weighting is simplified as follows:

$$p_w(\mathbf{f} \mid \mathbf{e}, a) = \prod_{i=1}^{n} \frac{1}{|\{j \mid (i,j) \in a\}|} \sum_{\forall (i,j) \in a} p(f_i \mid e_j)$$
$$\cong \prod_{i=1}^{n} \frac{1}{|\mathbf{e}|} \sum_{\forall e_j \in \mathbf{e}} p(f_i \mid e_j).$$

Then, it is trivial to compute the simplified lexical weighting of each MWE correspondence when the word translation probability table is provided. Here, we use the IBM model 1 to learn the table from the training data.

4.4 Evaluation Results

We trained a model using Moses toolkit (Koehn et al., 2007) on the training data as our baseline system.

Table 9 shows the influence of adding the MWE translations to the test data. In the first row (NIST06-sub), we only consider sentences containing MWE translations for BLEU score evaluation (726 sentences). In the second row, we took the whole NIST 2006 evaluation set into consideration (1,664 sentences). The Chinese words covered by the MWEs in NIST06-sub and NIST06 were 9.9% and 5.3% respectively.

Adding MWE translations to the test data statistically significantly lead to better results than those of the baseline. Significance was tested using a paired bootstrap (Koehn, 2004) with 1000 samples ($p<0.02$). Although the improvement in BLEU score seems small, it is actually reasonably good given that the MWEs account for only 5% of the NIST06 test set. Examples of improved translations are shown in Table 10. There is still room for improvement of the proposed MWE extraction method in order to provide more MWE translation pairs or design a feasible way to incorporate discontinuous bilingual MWEs to the decoder.

5 Conclusions and Future Work

We have proposed a high precision algorithm for extracting translations of multiword expressions from parallel corpora. The algorithm can be used to translate any language pair and any type of word sequence, including rigid sequences and discontinuous sequences. Our evaluation results show that the algorithm can cope with the difficulties caused by *indirect association* and the *common subsequence effects*, leading to significant improvement over the word alignment-based extraction methods used by the state of the art systems and other association-based extraction methods. We also demonstrate that extracted translations significantly improve the

[2] We set the threshold at 0.1.

performance of the Moses machine translation system.

In future work, it would be interesting to develop a machine translation model that can be integrated with the translation acquisition algorithm in a more effective way. Using the normalized-frequency score to help phrase alignment tasks, as the *grow-diag-final* heuristic, would also be interesting direction to explore.

Acknowledgement

This research was supported in part by the National Science Council of Taiwan under the NSC Grants: NSC 96-2221-E-001-023-MY3.

References

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263-311.

Gao, Jianfeng, Jian-Yun Nie, Hongzhao He, Weijun Chen, Ming Zhou. 2002. Resolving Query Translation Ambiguity using a Decaying Co-occurrence Model and Syntactic Dependence Relations. In *Proc. of SIGIR'02*. pp. 183 -190.

Kitamura, Mihoko and Yuji Matsumoto. 1996. Automatic Extraction of Word Sequence Correspondences in Parallel Corpora. In *Proc. of the 4th Annual Workshop on Very Large Corpora*. pp. 79-87.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proc. of HLT/NAACL'03*. pp. 127-133.

Koehn, Philipp. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP'04*. pp. 388-395.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL'07, demonstration session*.

Kupiec, Julian. 1993. An Algorithm for Finding Noun Phrase Correspondences in Bilingual Corpora. In *Proc. of ACL'93* . pp. 17-22.

Liang, Percy, Ben Taskar, Dan Klein. 2006. Alignment by Agreement. In *Proc. of HLT/NAACL'06*. pp. 104-111.

Ma, Yanjun, Nicolas Stroppa, Andy Way. 2007. Bootstrapping Word Alignment via Word Packing. In *Proc. of ACL'07*. pp. 304-311.

Ma, Yanjun, Sylwia Ozdowska, Yanli Sun, and Andy Way. 2008. Improving Word Alignment Using Syntactic Dependencies. In *Proc. of ACL/HLT'08 Second Workshop on Syntax and Structure in Statistical Translation.* pp. 69-77.

Melamed, Ilya Dan. 2001. Empirical Methods for Exploiting parallel Texts. *MIT press*.

Och, Franz Josef and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proc. of ACL'00*. pp. 440-447.

Och, Franz Josef, Christoph Tillmann, and Hermann Ney. 1999. Improved Alignment Models for Statistical Machine Translation. In *Proc. of EMNLP/VLC'99*. pp. 20-28.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proc. of ACL'02*. pp. 311-318.

Smadja, Frank, Kathleen R. McKeown, and Vasileios Hatzivassiloglou. 1996. Translating Collocations for Bilingual Lexicons: A Statistical Approach. *Computational Linguistics*, 22(1):1-38.

Wu, Dekai. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377-403.

Wu, Hua, Ming Zhou. 2003. Synonymous Collocation Extraction Using Translation Information. In *Proc. of ACL'03*. pp. 120-127.

Yamamoto, Kaoru, Yuji Matsumoto. 2000. Acquisition of Phrase-level Bilingual Correspondence using Dependency Structure. In *Proc. of COLING'00*. pp. 933-939.

Collocation Extraction Using Monolingual Word Alignment Method

Zhanyi Liu[1,2], Haifeng Wang[2], Hua Wu[2], Sheng Li[1]

[1]Harbin Institute of Technology, Harbin, China
[2]Toshiba (China) Research and Development Center, Beijing, China
{liuzhanyi,wanghaifeng,wuhua}@rdc.toshiba.com.cn
lisheng@hit.edu.cn

Abstract

Statistical bilingual word alignment has been well studied in the context of machine translation. This paper adapts the bilingual word alignment algorithm to monolingual scenario to extract collocations from monolingual corpus. The monolingual corpus is first replicated to generate a parallel corpus, where each sentence pair consists of two identical sentences in the same language. Then the monolingual word alignment algorithm is employed to align the potentially collocated words in the monolingual sentences. Finally the aligned word pairs are ranked according to refined alignment probabilities and those with higher scores are extracted as collocations. We conducted experiments using Chinese and English corpora individually. Compared with previous approaches, which use association measures to extract collocations from the co-occurring word pairs within a given window, our method achieves higher precision and recall. According to human evaluation in terms of precision, our method achieves absolute improvements of 27.9% on the Chinese corpus and 23.6% on the English corpus, respectively. Especially, we can extract collocations with longer spans, achieving a high precision of 69% on the long-span (>6) Chinese collocations.

1 Introduction

Collocation is generally defined as a group of words that occur together more often than by chance (McKeown and Radev, 2000). In this paper, a collocation is composed of two words occurring as either a consecutive word sequence or an interrupted word sequence in sentences, such as "by accident" or "take … advice". The collocations in this paper include phrasal verbs (e.g. "put on"), proper nouns (e.g. "New York"), idioms (e.g. "dry run"), compound nouns (e.g. "ice cream"), correlative conjunctions (e.g. "either … or"), and the other commonly used combinations in following types: verb+noun, adjective+noun, adverb+verb, adverb+adjective and adjective+preposition (e.g. "break rules", "strong tea", "softly whisper", "fully aware", and "fond of").

Many studies on collocation extraction are carried out based on co-occurring frequencies of the word pairs in texts (Choueka et al., 1983; Church and Hanks, 1990; Smadja, 1993; Dunning, 1993; Pearce, 2002; Evert, 2004). These approaches use association measures to discover collocations from the word pairs in a given window. To avoid explosion, these approaches generally limit the window size to a small number. As a result, long-span collocations can not be extracted[1]. In addition, since the word pairs in the given window are regarded as potential collocations, lots of false collocations exist. Although these approaches used different association measures to filter those false collocations, the precision of the extracted collocations is not high. The above problems could be partially solved by introducing more resources into collocation extraction, such as chunker (Wermter and Hahn, 2004), parser (Lin, 1998; Seretan and Wehrli, 2006) and WordNet (Pearce, 2001).

This paper proposes a novel *monolingual word alignment* (MWA) method to extract collocation of higher quality and with longer spans only from monolingual corpus, without using any additional resources. The difference between MWA and bilingual word alignment (Brown et al., 1993) is that the MWA method works on monolingual parallel corpus instead of bilingual corpus used by bilingual word alignment. The

[1] Here, "span of collocation" means the distance of two words in a collocation. For example, if the span of the collocation (w_1, w_2) is 6, it means there are 5 words interrupting between w_1 and w_2 in a sentence.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 487–495,
Singapore, 6-7 August 2009.

monolingual corpus is replicated to generate a parallel corpus, where each sentence pair consists of two identical sentences in the same language, instead of a sentence in one language and its translation in another language. We adapt the bilingual word alignment algorithm to the monolingual scenario to align the potentially collocated word pairs in the monolingual sentences, with the constraint that a word is not allowed to be aligned with itself in a sentence. In addition, we propose a ranking method to finally extract the collocations from the aligned word pairs. This method assigns scores to the aligned word pairs by using alignment probabilities multiplied by a factor derived from the exponential function on the frequencies of the aligned word pairs. The pairs with higher scores are selected as collocations.

The main contribution of this paper is that the well studied bilingual statistical word alignment method is successfully adapted to monolingual scenario for collocation extraction. Compared with the previous approaches, which use association measures to extract collocations, our method achieves much higher precision and slightly higher recall. The MWA method has the following three advantages. First, it explicitly models the co-occurring frequencies and position information of word pairs, which are integrated into a model to search for the potentially collocated word pairs in a sentence. Second, a new feature, *fertility*, is employed to model the number of words that a word can collocate with in a sentence. Finally, our method can obtain the long-span collocations. Human evaluations on the extracted Chinese collocations show that 69% of the long-span (>6) collocations are correct. Although the previous methods could also extract long-span collocations by setting the larger window size, the precision is very low.

In the remainder of this paper, Section 2 describes the MWA model for collocation extraction. Section 3 describes the initial experimental results. In Section 4, we propose a method to improve the MWA models. Further experiments are shown in Sections 5 and 6, followed by a discussion in Section 7. Finally, the conclusions are presented in Section 8.

2 Collocation Extraction With Monolingual Word Alignment Method

2.1 Monolingual Word Alignment

Given a bilingual sentence pair, a source language word can be aligned with its corresponding target language word. Figure 1 shows an example of Chinese-to-English word alignment.

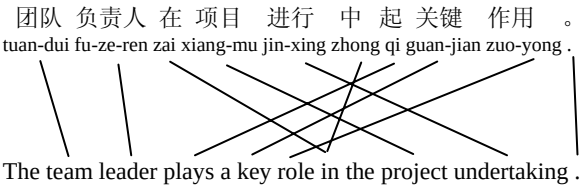
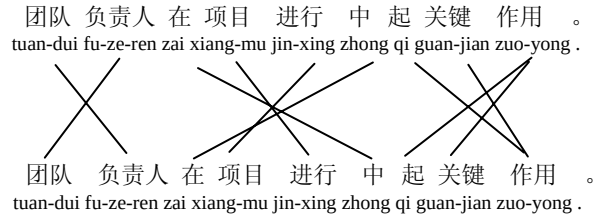


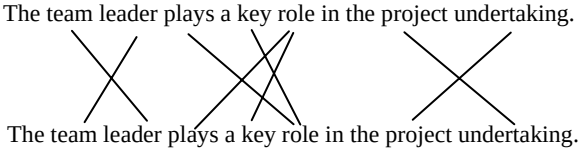
Figure 1. Bilingual word alignment

In Figure 1, a word in one language is aligned with its counterpart in the other language. For examples, the Chinese word "团队/tuan-dui" is aligned with its English translation "team", while the Chinese word "负责人/fu-ze-ren" is aligned with its English translation "leader".

In the Chinese sentence in Figure 1, there are some Chinese collocations, such as (团队/tuan-dui, 负责人/fu-ze-ren). There are also some English collocations in the English sentence, such as (team, leader). We separately illustrate the collocations in the Chinese sentence and the English sentence in Figure 2, where the collocated words are aligned with each other.



(a) Collocations in the Chinese sentence



(b) Collocations in the English sentence

Figure 2. Word alignments of collocations in sentence

Comparing the alignments in Figures 1 and 2, we can see that the task of monolingual collocations construction is similar to that of bilingual word alignment. In a bilingual sentence pair, a source word is aligned with its corresponding target word, while in a monolingual sentence, a word is aligned with its collocates. Therefore, it is reasonable to regard collocation construction as a task of aligning the collocated words in monolingual sentences.

Statistical bilingual word alignment method, which has been well studied in the context of machine translation, can extract the aligned bilingual word pairs from a bilingual corpus. This paper adapts the bilingual word alignment algorithm to monolingual scenario to align the collocated words in a monolingual corpus.

Given a sentence with l words $S = \{w_1, ..., w_l\}$, the word alignments $A = \{(i, a_i) \mid i \in [1, l]\}$ can be obtained by maximizing the word alignment probability of the sentence, according to Eq. (1).

$$A = \arg\max_{\forall A'} p(A' \mid S) \quad (1)$$

Where $(i, a_i) \in A$ means that the word w_i is aligned with the word w_{a_i}.

In a monolingual sentence, a word never collocates with itself. Thus the alignment set is denoted as $A = \{(i, a_i) \mid i \in [1, l] \,\&\, a_i \neq i\}$.

We adapt the bilingual word alignment model, IBM Model 3 (Brown et al., 1993), to monolingual word alignment. The probability of the alignment sequence is calculated using Eq. (2).

$$p(A \mid S) \propto \prod_{i=1}^{l} n(\phi_i \mid w_i) \prod_{j=1}^{l} t(w_j \mid w_{a_j}) d(j \mid a_j, l) \quad (2)$$

Where ϕ_i denotes the number of words that are aligned with w_i. Three kinds of probabilities are involved:

- Word collocation probability $t(w_j \mid w_{a_j})$, which describes the possibility of w_j collocating with w_{a_j};
- Position collocation probability $d(j, a_j, l)$, which describes the probability of a word in position a_j collocating with another word in position j;
- Fertility probability $n(\phi_i \mid w_i)$, which describes the probability of the number of words that a word w_i can collocate with (refer to subsection 7.1 for further discussion).

Figure 3 shows an example of word alignment on the English sentence in Figure 2 (b) with the MWA method. In the sentence, the 7th word "role" collocates with both the 4th word "play" and the 6th word "key". Thus, $t(w_4 \mid w_7)$ and $t(w_6 \mid w_7)$ describe the probabilities that the word "role" collocates with "play" and "key",

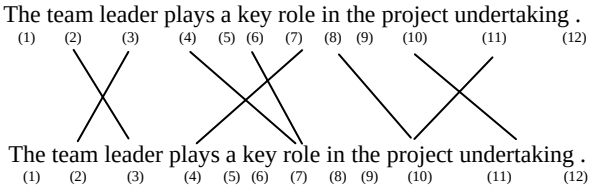


Figure 3. Results of MWA method

respectively. $d(4 \mid 7,12)$ and $d(6 \mid 7,12)$ describe the probabilities that the word in position 7 collocates with the words in position 4 and 6 in a sentence with 12 words. For the word "role", ϕ_7 is 2, which indicates that the word "role" collocates with two words in the sentence.

To train the MWA model, we implement a MWA tool for collocation extraction, which uses similar training methods for bilingual word alignment, except that a word can not be aligned to itself.

2.2 Collocation Extraction

Given a monolingual corpus, we use the trained MWA model to align the collocated words in each sentence. As a result, we can generate a set of aligned word pairs on the corpus. According to the alignment results, we calculate the frequency for two words aligned in the corpus, denoted as $freq(w_i, w_j)$. In our method, we filtered those aligned word pairs whose frequencies are lower than 5. Based on the alignment frequency, we estimate the alignment probabilities for each aligned word pair as shown in Eq. (3) and (4).

$$p(w_i \mid w_j) = \frac{freq(w_i, w_j)}{\sum_{w'} freq(w', w_j)} \quad (3)$$

$$p(w_j \mid w_i) = \frac{freq(w_i, w_j)}{\sum_{w'} freq(w_i, w')} \quad (4)$$

With alignment probabilities, we assign scores to the aligned word pairs and those with higher scores are selected as collocations, which are estimated as shown in Eq. (5).

$$\bar{p}(w_i, w_j) = \frac{p(w_i \mid w_j) + p(w_j \mid w_i)}{2} \quad (5)$$

3 Initial Experiments

In this experiment, we used the method as described in Section 2 for collocation extraction. Since our method does not use any linguistic information, we compared our method with the

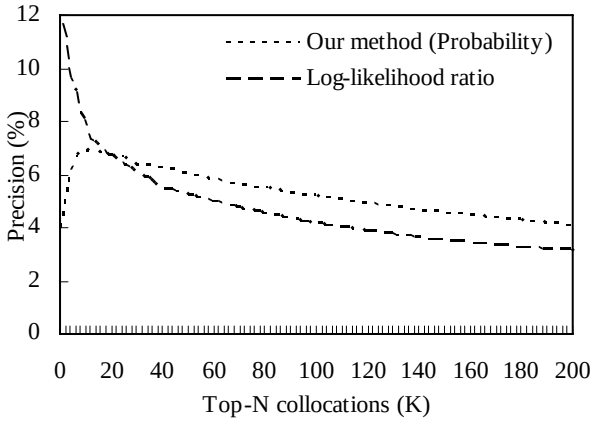


Figure 4. Precision of collocations

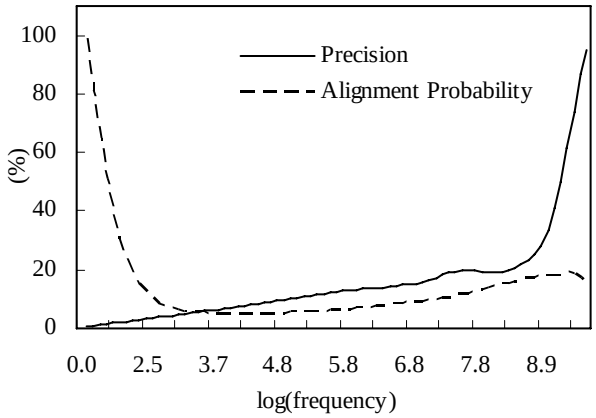


Figure 5. Frequency vs. precision/alignment probability

baseline methods without using linguistic knowledge. These baseline methods take all co-occurring word pairs within a given window as collocation candidates, and then use association measures to rank the candidates. Those candidates with higher association scores are extracted as collocations. In this paper, the window size is set to [-6, +6].

3.1 Data

The experiments were carried out on a Chinese corpus, which consists of one year (2004) of the Xinhua news corpus from LDC[2], containing about 28 millions of Chinese words. Since punctuations are rarely used to construct collocations, they were removed from the corpora. To automatically estimate the precision of extracted collocations on the Chinese corpus, we built a gold set by collecting Chinese collocations from handcrafted collocation dictionaries, containing 56,888 collocations.

3.2 Results

The precision is automatically calculated against the gold set according to Eq. (6).

$$precision = \frac{\#(C_{\text{Top}-N} \bigcap C_{\text{gold}})}{\#(C_{\text{Top}-N})} \qquad (6)$$

Where $C_{\text{Top-}N}$ and C_{gold} denote the top collocations in the N-best list and the collocations in the gold set, respectively.

We compared our method with several baseline methods using different association measures[3]: co-occurring frequency, log-likelihood ratio, chi-square test, mutual information, and t-test. Among them, the log-likelihood ratio measure achieves the best performance. Thus, in this paper, we only show the performance of the log-likelihood ratio measure.

Figure 4 shows the precisions of the top N collocations as N steadily increases with an increment of 1K, which are extracted by our method and the baseline method using log-likelihood ratio as the association measure.

The absolute precision of collocations is not high in the figure. For example, among the top 200K collocations, about 4% of the collocations are correct. This is because our gold set contains only about 57K collocations. Even if all collocations in the gold set are included in the 200K-best list, the precision is only 28%. Thus, it is more useful to compare precision curves for collocations in the N-best lists extracted by different methods. In addition, since this gold set only includes a small number of collocations, the precision curves of our method and the baseline method are getting closer, as N increases. For example, when N is set to 200K, our method and the baseline method achieved precisions of 4.09% and 3.12%, respectively. And when N is set to 400K, they achieved 2.78% and 2.26%, respectively. For convenience of comparison, we set N up to 200K in the experiments.

From the results, it can also be seen that, among the N-best lists with N less than 20K, the precision of the collocations extracted by our method is lower than that of the collocations extracted by the baseline, and became higher when N is larger than 20K.

In order to analyze the possible reasons, we investigated the relationships among the frequencies of the aligned word pairs, the alignment

[2] Available at: http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2007T03

[3] The definitions of these measures can be found in Manning and Schütze (1999).

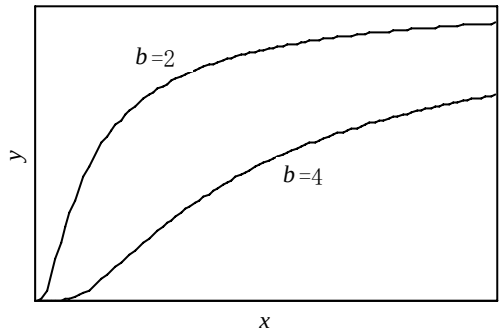


Figure 6. $y = e^{-b/x}$

probabilities, and precisions of collocations, which are shown in Figure 5. From the figure, we can see (1) that the lower the frequencies of the aligned word pairs are, the higher the alignment probabilities are; and (2) that the precisions of the aligned word pairs with lower frequencies is lower. According to the above observations, we conclude that it is the word pairs with lower frequencies but higher probabilities that caused the lower precision of the top 20K collocations extracted by our method.

4 Improved MWA Method

According to the analysis in subsection 3.2, we need to penalize the aligned word pairs with lower frequencies. In order to achieve the above goal, we need to refine the alignment probabilities by using a penalization factor derived from a function on the frequencies of the aligned word pairs. This function $y = f(x)$ should satisfy the following two conditions, where x represents the log function of frequencies.

(1) The function is monotonic. When x is set to a smaller number, y is also small. This results in the penalization on the aligned word pairs with lower frequencies.
(2) When $x \to \infty$, y is set to 1. This means that we don't penalize the aligned word pairs with higher frequencies.

According to the above descriptions, we propose to use the exponential function in Eq. (7).

$$y = e^{-b/x} \tag{7}$$

Figure 6 describes this function. The constant b in the function is used to adjust the shape of the line. The line is sharp with b set to a small number, while the line is flat with b set to a larger number. In our case, if b is set to a larger number, we assign a larger penalization weight to those aligned word pairs with lower frequencies.

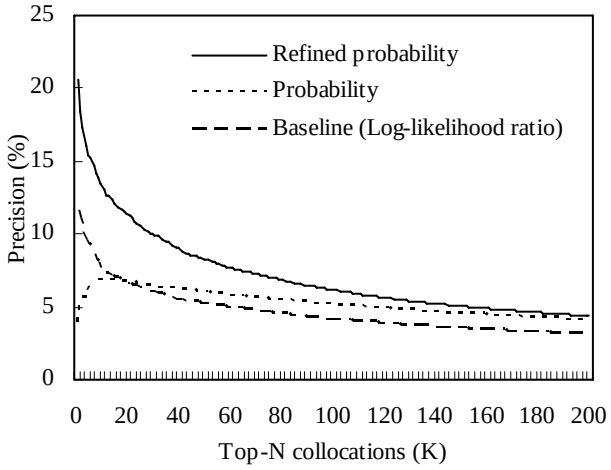


Figure 7. Precision of collocations extracted by the improved method

According to the above discussion, we can use the following measure to assign scores to the aligned words pairs generated by the MWA method.

$$\overline{p}_r(w_i, w_j) = \frac{p(w_i \mid w_j) + p(w_j \mid w_i)}{2} \times e^{-\frac{b}{\log(freq(w_i, w_j))}} \tag{8}$$

Where w_i and w_j are two aligned words. $p(w_i|w_j)$ and $p(w_j|w_i)$ are alignment probabilities as shown in Eq. (3) and (4). $\log(freq(w_i, w_j))$ is the log function of the frequencies of the aligned word pairs (w_i, w_j).

5 Evaluation on Chinese corpus

We used the same Chinese corpus described in Section 3 to evaluate the improved method as shown in Section 4. In the experiments, b was tuned by using a development set and set to 25.

5.1 Precision

In this section, we evaluated the extracted collocations in terms of precision using both automatic evaluation and human evaluation.

Automatic Evaluation

Figure 7 shows the precisions of the collocations in the N-best lists extracted by our method and the baseline method against the gold set in Section 3. For our methods, we used two different measures to rank the aligned word pairs: alignment probabilities in Eq. (5) and refined

		Our method	Baseline
True		569	290
False	A	25	16
	B	5	4
	C	240	251
	D	161	439

Table 1. Manual evaluation of the top 1K Chinese collocations. The precisions of our method and the baseline method are 56.9% and 29.0%, respectively.

alignment probabilities in Eq. (8). From the results, it can be seen that with the refined alignment probabilities, our method achieved the highest precision on the N-best lists, which greatly outperforms the best baseline method. For example, in the top 1K list, our method achieves a precision of 20.6%, which is much higher than the precision of the baseline method (11.7%). This indicates that the exponential function used to penalize the alignment probabilities plays a key role in demoting most of the aligned word pairs with low frequencies.

Human Evaluation

In automatic evaluation, the gold set only contains collocations in the existing dictionaries. Some collocations related to specific corpora are not included in the set. Therefore, we selected the top 1K collocations extracted by our improved method to manually estimate the precision. During human evaluation, the true collocations are denoted as "**True**" in our experiments. The false collocations were further classified into the following classes.

A: The candidate consists of two words that are semantically related, such as (医生 doctor, 护士 nurse).

B: The candidate is a part of the multi-word (≥3) collocation. For example, (自我 self, 机制 mechanism) is a part of the three-word collocation (自我 self, 约束 regulating, 机制 mechanism).

C: The candidates consist of the adjacent words that frequently occur together, such as (他 he, 说 say) and (很 very, 好 good).

D: Two words in the candidates have no relationship with each other, but occur together frequently, such as (北京 Beijing, 月 month) and (和 and, 为 for).

Table 1 shows the evaluation results. Our method extracted 569 true collocations, which

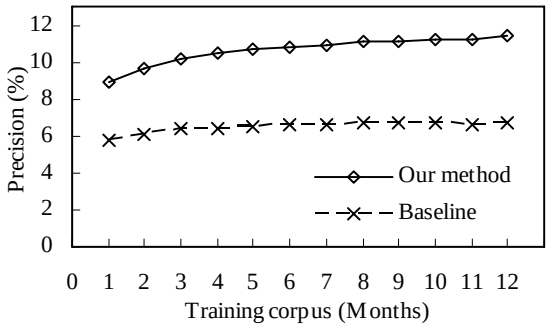


Figure 8. Corpus size vs. precision

are much more than those extracted by the baseline method. Further analysis shows that, in addition to extracting short-span collocations, our method extracted collocations with longer spans as compared with the baseline method. For example, (处于 in, 状态 state) and (由于 because, 因此 so) are two long-span collocations. Among the 1K collocations, there are 48 collocation candidates whose spans are larger than 6, which are not covered by the baseline method since the window size is set to 6. And 33 of them are true collocations, with a higher precision of 69%.

Classes C and D account for the most part of the false collocations. Although the words in these two classes co-occur frequently, they can not be regarded as collocations. And we also found out that the errors in class D produced by the baseline method are much more than that of those produced by our method. This indicates that our MWA method can remove much more noise from the frequently occurring word pairs.

In Class A, the two words are semantically related and occur together in the corpus. These kinds of collocations can not be distinguished from the true collocations by our method without additional resources.

Since only bigram collocations were extracted by our method, the multi-word (≥3) collocations were split into bigram collocations, which caused the error collocations in Class B[4].

Corpus size vs. precision

Here, we investigated the effect of the corpus size on the precision of the extracted collocations. We evaluated the precision against the gold set as shown in the automatic evaluation. First, the whole corpus (one year of newspaper) was split into 12 parts according to the published months. Then we calculated the precisions as the training

[4] Since only a very small faction of collocations contain more than two words, a few error collocations belong to Class B.

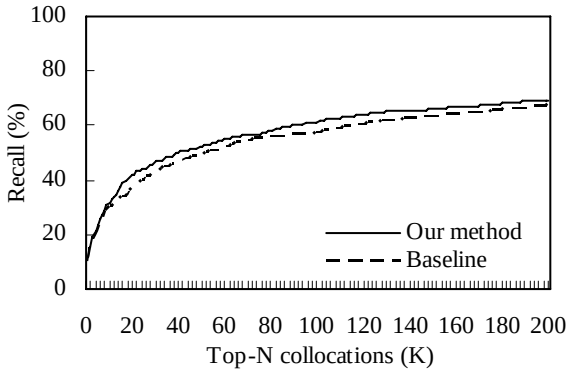


Figure 9. Recall on the Chinese corpus

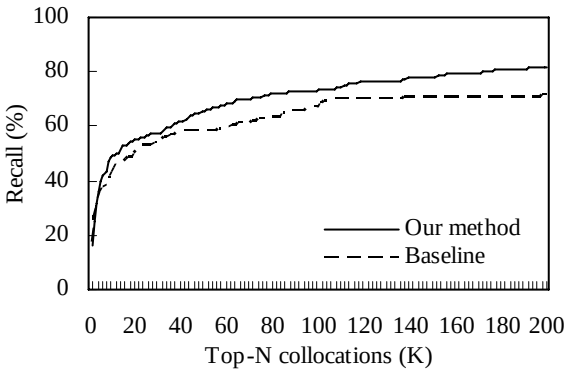


Figure 10. Recall on the English corpus

corpus increases part by part. The top 20K collocations were selected for evaluation.

Figure 8 shows the experimental results. The precision of collocations extracted by our method is obviously higher than that of collocations extracted by the baseline method. When the size of the training corpus became larger, the difference between our method and the baseline method also became bigger. When the training corpus contains more than 9 months of corpora, the precision of collocations extracted by the baseline method did not increase anymore. However, the precision of collocations extracted by our method kept on increasing. This indicates the MWA method can extract more true collocations of higher quality when it is trained with larger size of training data.

5.2 Recall

Recall was evaluated on a manually labeled subset of the training corpus. The subset contains 100 sentences that were randomly selected from the whole corpus. The sentence average length is 24. All true collocations (660) were labeled manually. The recall was calculated according to Eq. (9).

$$recall = \frac{\#(C_{\text{Top}-N} \bigcap C_{\text{subset}})}{\#(C_{\text{subset}})} \qquad (9)$$

Here, $C_{\text{Top-}N}$ denotes the top collocations in the N-best list and C_{subset} denotes the true collocations in the subset.

Figure 9 shows the recalls of collocations extracted by our method and the baseline method on the labeled subset. The results show that our method can extract more true collocations than the baseline method.

		Our method	Baseline
True		591	355
False	A	11	4
	B	19	20
	C	200	136
	D	179	485

Table 2. Manual evaluation of the top 1K English collocations. The precisions of our method and the baseline method are 59.1% and 35.5%, respectively.

In our experiments, the baseline method extracts about 20 millions of collocation candidates, while our method only extracts about 3 millions of collocation candidates[5]. Although the collocations of our method are much less than that of the baseline, the experiments show that the recall of our method is higher. This again proved that our method has the stronger ability to distinguish true collocations from false collocations.

6 Evaluation on English corpus

We also manually evaluated the proposed method on an English corpus, which is a subset randomly extracted from the British National Corpus[6]. The English corpus contains about 20 millions of words.

6.1 Precision

We estimated the precision of the top 1K collocations. Table 2 shows the results. The classification of the false collocations is the same as that in Table 1. The results show that our methods outperformed the baseline method using log-

[5] We set the threshold to 7.88 with a confidence level of $\alpha = 0.005$ (cf. page 174 of Chapter 5 in (McKeown and Radev, 2000) for more details).

[6] Available at: http://www.hcu.ox.ac.uk/BNC/

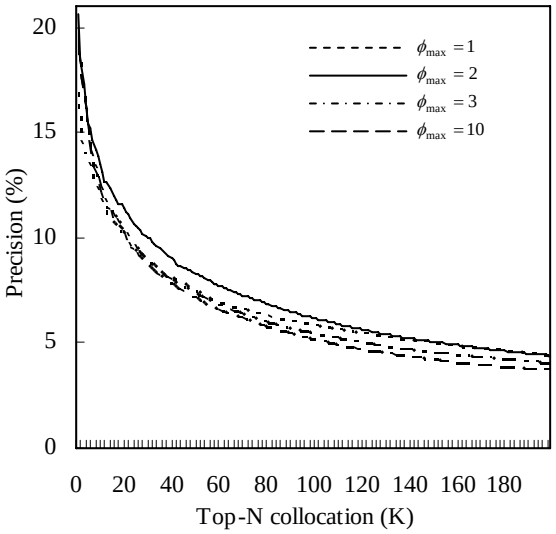


Figure 11. Fertility vs. precision

likelihood ratio. And the distribution of the false collocations is similar to that on the Chinese corpus.

6.2 Recall

We used the method described in subsection 5.2 to calculate the recall. 100 English sentences were labeled manually, obtaining 205 true collocations. Figure 10 shows the recall of the collocations in the N-best lists. From the figure, it can be seen that the trend on the English corpus is similar to that on the Chinese corpus, which indicates that our method is language-independent.

7 Discussion

7.1 The Effect of Fertility

In the MWA model as described in subsection 2.1, ϕ_i denotes the number of words that can align with w_i. Since a word only collocates with a few other words in a sentence, we should set a maximum number for ϕ, denote as $\phi_{\max}$.

In order to set $\phi_{\max}$, we examined the true collocations in the manually labeled set described in subsection 5.2. We found that 78% of words collocate with only one word, and 17% of words collocate with two words. In sum, 95% of words in the corpus can only collocate with at most two words. According to the above observation, we set $\phi_{\max}$ to 2.

In order to further examine the effect of $\phi_{\max}$ on collocation extraction, we used several different $\phi_{\max}$ in our experiments. The comparison

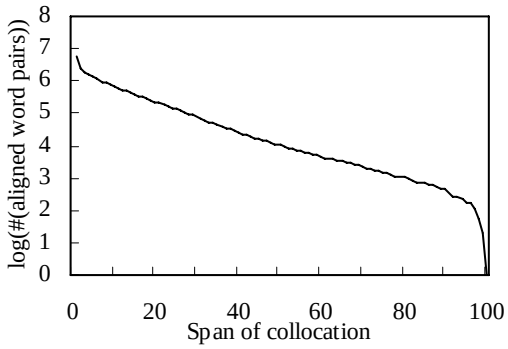


Figure 12. Distribution of spans

results are shown in Figure 11. The highest precision is achieved when $\phi_{\max}$ is set to 2. This result verifies our observation on the corpus.

7.2 Span of Collocation

One of the advantages of our method is that long-span collocations can be reliably extracted. In this subsection, we investigate the distribution of the span of the aligned word pairs. For the aligned word pairs occurring more than once, we calculated the average span as shown in Eq. (10).

$$AveSpan(w_i, w_j) = \frac{\sum_{s \in corpus} Span(w_i, w_j; s)}{freq(w_i, w_j)} \quad (10)$$

Where, $Span(w_i, w_j; s)$ is the span of the words w_i and w_j in the sentence s; $AveSpan(w_i, w_j)$ is the average span.

The distribution is shown in Figure 12. It can be seen that the number of the aligned word pairs decreased exponentially as the average span increased. About 17% of the aligned word pairs have spans longer than 6. According to the human evaluation result for precision in subsection 5.1, the precision of the long-span collocations is even higher than that of the short-span collocations. This indicates that our method can extract reliable collocations with long spans.

8 Conclusion

We have presented a monolingual word alignment method to extract collocations from monolingual corpus. We first replicated the monolingual corpus to generate a parallel corpus, in which each sentence pair consists of the two identical sentences in the same language. Then we adapted the bilingual word alignment algorithm to the monolingual scenario to align the

potentially collocated word pairs in the monolingual sentences. In addition, a ranking method was proposed to finally extract the collocations from the aligned word pairs. It scores collocation candidates by using alignment probabilities multiplied by a factor derived from the exponential function on the frequencies. Those with higher scores are selected as collocations. Both Chinese and English collocation extraction experiments indicate that our method outperforms previous approaches in terms of both precision and recall. For example, according to the human evaluations on the Chinese corpus, our method achieved a precision of 56.9%, which is much higher than that of the baseline method (29.0%). Moreover, we can extract collocations with longer span. Human evaluation on the extracted Chinese collocations shows that 69% of the long-span (>6) collocations are correct.

References

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2): 263-311.

Yaacov Choueka, S.T. Klein, and E. Neuwitz. 1983. Automatic Retrieval of Frequent Idiomatic and Collocational Expressions in a Large Corpus. *Journal for Literary and Linguistic computing*, 4(1):34-38.

Kenneth Church and Patrick Hanks. 1990. Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics*, 16(1):22-29.

Ted Dunning. 1993. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1): 61-74.

Stefan Evert. 2004. The Statistics of Word Cooccurrences: Word Pairs and Collocations. *Ph.D. thesis*, University of Stuttgart.

Dekang Lin. 1998. Extracting Collocations from Text Corpora. In *Proceedings of the 1st Workshop on Computational Terminology*, pp. 57-63.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*, Cambridge, MA; London, U.K.: Bradford Book & MIT Press.

Kathleen R. McKeown and Dragomir R. Radev. 2000. Collocations. In Robert Dale, Hermann Moisl, and Harold Somers (Ed.), *A Handbook of Natural Language Processing*, pp. 507-523.

Darren Pearce. 2001. Synonymy in Collocation Extraction. In *Proceedings of NAACL-2001 Workshop on Wordnet and Other Lexical Resources: Applications, Extensions and Customizations*, pp. 41-46.

Darren Pearce. 2002. A Comparative Evaluation of Collocation Extraction Techniques. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pp. 651-658.

Violeta Seretan and Eric Wehrli. 2006. Accurate Collocation Extraction Using a Multilingual Parser. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics* (*COLING/ACL-2006*), pp. 953-960

Frank Smadja. 1993. Retrieving Collocations from Text: Xtract. *Computational Linguistics*, 19(1): 143-177.

Joachim Wermter and Udo Hahn. 2004. Collocation Extraction Based on Modifiability Statistics. In *Proceedings of the 20th International Conference on Computational Linguistics* (*COLING-2004*), pp. 980-986.

Multi-Class Confidence Weighted Algorithms

Koby Crammer[*] **Mark Dredze**[†] **Alex Kulesza**[*]
[*]Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{crammer,kulesza}@cis.upenn.edu

[†]Human Language Technology Center of Excellence
Johns Hopkins University
Baltimore, MD 21211
mdredze@cs.jhu.edu

Abstract

The recently introduced online confidence-weighted (CW) learning algorithm for binary classification performs well on many binary NLP tasks. However, for multi-class problems CW learning updates and inference cannot be computed analytically or solved as convex optimization problems as they are in the binary case. We derive learning algorithms for the multi-class CW setting and provide extensive evaluation using nine NLP datasets, including three derived from the recently released New York Times corpus. Our best algorithm outperforms state-of-the-art online and batch methods on eight of the nine tasks. We also show that the confidence information maintained during learning yields useful probabilistic information at test time.

1 Introduction

Online learning algorithms such as the Perceptron process one example at a time, yielding simple and fast updates. They generally make few statistical assumptions about the data and are often used for natural language problems, where high dimensional feature representations, e.g., bags-of-words, demand efficiency. Most online algorithms, however, do not take into account the unique properties of such data, where many features are extremely rare and a few are very frequent.

Dredze, Crammer and Pereira (Dredze et al., 2008; Crammer et al., 2008) recently introduced confidence weighted (CW) online learning for binary prediction problems. CW learning explicitly models classifier weight uncertainty using a multivariate Gaussian distribution over weight vectors. The learner makes online updates based on its confidence in the current parameters, making larger changes in the weights of infrequently observed features. Empirical evaluation has demonstrated the advantages of this approach for a number of binary natural language processing (NLP) problems.

In this work, we develop and test multi-class confidence weighted online learning algorithms. For binary problems, the update rule is a simple convex optimization problem and inference is analytically computable. However, neither is true in the multi-class setting. We discuss several efficient online learning updates. These update rules can involve one, some, or all of the competing (incorrect) labels. We then perform an extensive evaluation of our algorithms using nine multi-class NLP classification problems, including three derived from the recently released New York Times corpus (Sandhaus, 2008). To the best of our knowledge, this is the first learning evaluation on these data. Our best algorithm outperforms state-of-the-art online algorithms and batch algorithms on eight of the nine datasets.

Surprisingly, we find that a simple algorithm in which updates consider only a single competing label often performs as well as or better than multi-constraint variants if it makes multiple passes over the data. This is especially promising for large datasets, where the efficiency of the update can be important. In the true online setting, where only one iteration is possible, multi-constraint algorithms yield better performance.

Finally, we demonstrate that the label distributions induced by the Gaussian parameter distributions resulting from our methods have interesting properties, such as higher entropy, compared to those from maximum entropy models. Improved label distributions may be useful in a variety of learning settings.

2 Problem Setting

In the multi-class setting, instances from an input space $\mathcal{X}$ take labels from a finite set $\mathcal{Y}$, $|\mathcal{Y}| = K$.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 496–504,
Singapore, 6-7 August 2009.

We use a standard approach (Collins, 2002) for generalizing binary classification and assume a feature function $\boldsymbol{f}(\boldsymbol{x}, y) \in \mathbb{R}^d$ mapping instances $\boldsymbol{x} \in \mathcal{X}$ and labels $y \in \mathcal{Y}$ into a common space.

We work in the online framework, where learning is performed in rounds. On each round the learner receives an input $\boldsymbol{x}_i$, makes a prediction $\hat{y}_i$ according to its current rule, and then learns the true label y_i. The learner uses the new example $(\boldsymbol{x}_i, y_i)$ to modify its prediction rule. Its goal is to minimize the total number of rounds with incorrect predictions, $|\{i : y_i \neq \hat{y}_i\}|$.

In this work we focus on linear models parameterized by weights $\boldsymbol{w}$ and utilizing prediction functions of the form $h_{\boldsymbol{w}}(\boldsymbol{x}) = \arg\max_z \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, z)$. Note that since we can choose $\boldsymbol{f}(\boldsymbol{x}, y)$ to be the vectorized Cartesian product of an input feature function $\boldsymbol{g}(\boldsymbol{x})$ and y, this setup generalizes the use of unique weight vectors for each element of $\mathcal{Y}$.

3 Confidence Weighted Learning

Dredze, Crammer, and Pereira (2008) introduced online confidence weighted (CW) learning for binary classification, where $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{\pm 1\}$. Rather than using a single parameter vector $\boldsymbol{w}$, CW maintains a distribution over parameters $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ the multivariate normal distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$. Given an input instance $\boldsymbol{x}$, a Gibbs classifier draws a weight vector $\boldsymbol{w}$ from the distribution and then makes a prediction according to the sign of $\boldsymbol{w} \cdot \boldsymbol{x}$.

This prediction rule is robust if the example is classified correctly with high-probability, that is, for some confidence parameter $.5 \leq \eta < 1$, $\Pr_{\boldsymbol{w}}[y(\boldsymbol{w} \cdot \boldsymbol{x}) \geq 0] \geq \eta$. To learn a binary CW classifier in the online framework, the robustness property is enforced at each iteration while making a minimal update to the parameter distribution in the KL sense:

$$
\begin{aligned}
(\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}) = & \\
& \arg\min_{\boldsymbol{\mu}, \Sigma} \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \,\|\, \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)\right) \\
& \text{s.t. } \Pr_{\boldsymbol{w}}[y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i) \geq 0] \geq \eta
\end{aligned} \tag{1}
$$

Dredze et al. (2008) showed that this optimization can be solved in closed form, yielding the updates

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \alpha_i \Sigma_i \boldsymbol{x}_i \tag{2}$$

$$\Sigma_{i+1} = \left(\Sigma_i^{-1} + \beta_i \boldsymbol{x}_i \boldsymbol{x}_i^T\right)^{-1} \tag{3}$$

for appropriate α_i and β_i.

For prediction, they use the Bayesian rule

$$\hat{y} = \arg\max_{z \in \{\pm 1\}} \Pr_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)}[z(\boldsymbol{x} \cdot \boldsymbol{w}) \geq 0] \ ,$$

which for binary labels is equivalent to using the mean parameters directly, $\hat{y} = \mathrm{sign}(\boldsymbol{\mu} \cdot \boldsymbol{x})$.

4 Multi-Class Confidence Weighted Learning

As in the binary case, we maintain a distribution over weight vectors $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$. Given an input instance $\boldsymbol{x}$, a Gibbs classifier draws a weight vector $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ and then predicts the label with the maximal score, $\arg\max_z (\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, z))$. As in the binary case, we use this prediction rule to define a robustness condition and corresponding learning updates.

We generalize the robustness condition used in Crammer et al. (2008). Following the update on round i, we require that the ith instance is correctly labeled with probability at least $\eta < 1$. Among the distributions that satisfy this condition, we choose the one that has the minimal KL distance from the current distribution. This yields the update

$$
\begin{aligned}
(\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}) = & \\
& \arg\min_{\boldsymbol{\mu}, \Sigma} \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \,\|\, \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)\right) \\
& \text{s.t. } \Pr[y_i \,|\, \boldsymbol{x}_i, \boldsymbol{\mu}, \Sigma] \geq \eta \ ,
\end{aligned} \tag{4}
$$

where

$$
\begin{aligned}
& \Pr[y \,|\, \boldsymbol{x}, \boldsymbol{\mu}, \Sigma] = \\
& \Pr_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)}\left[y = \arg\max_{z \in \mathcal{Y}} (\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, z))\right] \ .
\end{aligned}
$$

Due to the $\max$ operator in the constraint, this optimization is not convex when $K > 2$, and it does not permit a closed form solution. We therefore develop approximations that can be solved efficiently. We define the following set of events for a general input $\boldsymbol{x}$:

$$
\begin{aligned}
A_{r,s}(\boldsymbol{x}) &\stackrel{\text{def}}{=} \{\boldsymbol{w} : \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, r) \geq \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, s)\} \\
B_r(\boldsymbol{x}) &\stackrel{\text{def}}{=} \{\boldsymbol{w} : \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, r) \geq \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, s) \ \ \forall s\} \\
&= \bigcap_{s \neq r} A_{r,s}(\boldsymbol{x})
\end{aligned}
$$

We assume the probability that $\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, r) = \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, s)$ for some $s \neq r$ is zero, which

holds for non-trivial distribution parameters and feature vectors. We rewrite the prediction $\hat{y} = \arg\max_r \Pr[B_r(\boldsymbol{x})]$, and the constraint from Eq. (4) becomes

$$\Pr[B_{y_i}(\boldsymbol{x})] \geq \eta \,. \tag{5}$$

We focus now on approximating the event $B_{y_i}(\boldsymbol{x})$ in terms of events $A_{y_i,r}$. We rely on the fact that the level sets of $\Pr[A_{y_i,r}]$ are convex in $\boldsymbol{\mu}$ and Σ. This leads to convex constraints of the form $\Pr[A_{y_i,r}] \geq \gamma$.

Outer Bound: Since $B_r(\boldsymbol{x}) \subseteq A_{r,s}(\boldsymbol{x})$, it holds trivially that $\Pr[B_{y_i}(\boldsymbol{x})] \geq \eta \Rightarrow \Pr[A_{y_i,r}] \geq \eta, \forall r \neq y_i$. Thus we can replace the constraint $\Pr[B_{y_i}(\boldsymbol{x})] \geq \eta$ with $\Pr[A_{y_i,r}] \geq \eta$ to achieve an outer bound. We can simultaneously apply *all* of the pairwise constraints to achieve a tighter bound:

$$\Pr[A_{y_i,r}] \geq \eta \quad \forall r \neq y_i$$

This yields a convex approximation to Eq. (4) that may improve the objective value at the cost of violating the constraint. In the context of learning, this means that the new parameter distribution will be close to the previous one, but may not achieve the desired confidence on the current example. This makes the updates more conservative.

Inner Bound: We can also consider an inner bound. Note that $B_{y_i}(\boldsymbol{x})^c = (\cap_r A_{y_i,r}(\boldsymbol{x}))^c = \cup_r A_{y_i,r}(\boldsymbol{x})^c$, thus the constraint $\Pr[B_{y_i}(\boldsymbol{x})] \geq \eta$ is equivalent to

$$\Pr[\cup_r A_{y_i,r}(\boldsymbol{x})^c] \leq 1 - \eta \,,$$

and by the union bound, this follows whenever

$$\sum_r \Pr[A_{y_i,r}(\boldsymbol{x})^c] \leq 1 - \eta \,.$$

We can achieve this by choosing non-negative $\zeta_r \geq 0$, $\sum_r \zeta_r = 1$, and constraining

$$\Pr[A_{y_i,r}(\boldsymbol{x})] \geq 1 - (1-\eta)\,\zeta_r \quad \text{for} \quad r \neq y_i \,.$$

This formulation yields an inner bound on the original constraint, guaranteeing its satisfaction while possibly increasing the objective. In the context of learning, this is a more aggressive update, ensuring that the current example is robustly classified even if doing so requires a larger change to the parameter distribution.

Algorithm 1 Multi-Class CW Online Algorithm

Input: Confidence parameter η
Feature function $\boldsymbol{f}(\boldsymbol{x}, y) \in \mathbb{R}^d$
Initialize: $\boldsymbol{\mu}_1 = \mathbf{0}$, $\Sigma_1 = I$
for $i = 1, 2 \ldots$ **do**
Receive $\boldsymbol{x}_i \in \mathcal{X}$
Predict ranking of labels $\hat{y}_1, \hat{y}_2, \ldots$
Receive $y_i \in \mathcal{Y}$
Set $\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}$ by approximately solving Eq. (4) using one of the following:
Single-constraint update (Sec. 5.1)
Exact many-constraint update (Sec. 5.2)
Seq. many-constraint approx. (Sec. 5.2)
Parallel many-constraint approx. (Sec. 5.2)
end for
Output: Final $\boldsymbol{\mu}$ and Σ

Discussion: The two approximations are quite similar in form. Both replace the constraint $\Pr[B_{y_i}(\boldsymbol{x})] \geq \eta$ with one or more constraints of the form

$$\Pr[A_{y_i,r}(\boldsymbol{x})] \geq \eta_r \,. \tag{6}$$

To achieve an outer bound we choose $\eta_r = \eta$ for any set of $r \neq y_i$. To achieve an inner bound we use all $K-1$ possible constraints, setting $\eta_r = 1 - (1-\eta)\,\zeta_r$ for suitable ζ_r. A simple choice is $\zeta_r = 1/(K-1)$.

In practice, η is a learning parameter whose value will be optimized for each task. In this case, the outer bound (when all constraints are included) and inner bound (when $\zeta_r = 1/(K-1)$) can be seen as equivalent, since for any fixed value of $\eta^{(\text{in})}$ for the inner bound we can choose

$$\eta^{(\text{out})} = 1 - \frac{1-\eta^{(\text{in})}}{K-1} \,,$$

for the outer bound and the resulting η_r will be equal. By optimizing η we automatically tune the approximation to achieve the best compromise between the inner and outer bounds. In the following, we will therefore assume $\eta_r = \eta$.

5 Online Updates

Our algorithms are online and process examples one at a time. Pseudo-code for our approach is given in algorithm 1. We approximate the prediction step by ranking each label y according to the score given by the mean weight vector, $\boldsymbol{\mu} \cdot \boldsymbol{f}(\boldsymbol{x}_i, y)$. Although this approach is Bayes optimal for binary problems (Dredze et al., 2008),

it is an approximation in general. We note that more accurate inference can be performed in the multi-class case by sampling weight vectors from the distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ or selecting labels sensitive to the variance of prediction; however, in our experiments this did not improve performance and required significantly more computation. We therefore proceed with this simple and effective approximation.

The update rule is given by an approximation of the type described in Sec. 4. All that remains is to choose the constraint set and solve the optimization efficiently. We discuss several schemes for minimizing KL divergence subject to one or more constraints of the form $\Pr[A_{y_i,r}(\boldsymbol{x})] \geq \eta$. We start with a single constraint.

5.1 Single-Constraint Updates

The simplest approach is to select the single constraint $\Pr[A_{y_i,r}(\boldsymbol{x})] \geq \eta$ corresponding to the highest-ranking label $r \neq y_i$. This ensures that, following the update, the true label is more likely to be predicted than the label that was its closest competitor. We refer to this as the $k = 1$ update.

Whenever we have only a single constraint, we can reduce the optimization to one of the closed-form CW updates used for binary classification. Several have been proposed, based on linear approximations (Dredze et al., 2008) and exact formulations (Crammer et al., 2008). For simplicity, we use the `Variance` method from Dredze et al. (2008), which did well in our initial evaluations. This method leads to the following update rules. Note that in practice Σ is projected to a diagonal matrix as part of the update; this is necessary due to the large number of features that we use.

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \alpha_i \Sigma_i \boldsymbol{g}_{i,y_i,r} \tag{7}$$

$$\Sigma_{i+1} = \left(\Sigma_i^{-1} + 2\alpha_i \phi \boldsymbol{g}_{i,y_i,r} \boldsymbol{g}_{i,y_i,r}^\top\right)^{-1} \tag{8}$$

$$\boldsymbol{g}_{i,y_i,r} = \boldsymbol{f}(\boldsymbol{x}_i, y_i) - \boldsymbol{f}(\boldsymbol{x}_i, r) \quad \phi = \Phi^{-1}(\eta)$$

The scale α_i is given by $\max(\gamma_i, 0)$, where γ_i is equal to

$$\frac{-(1 + 2\phi m_i) + \sqrt{(1 + 2\phi m_i)^2 - 8\phi(m_i - \phi v_i)}}{4\phi v_i}$$

and

$$m_i = \boldsymbol{\mu}_i \cdot \boldsymbol{g}_{i,y_i,r} \qquad v_i = \boldsymbol{g}_{i,y_i,r}^\top \Sigma_i \boldsymbol{g}_{i,y_i,r} \,.$$

These rules derive directly from Dredze et al. (2008) or Figure 1 in Crammer et al. (2008); we simply substitute $y_i = 1$ and $\boldsymbol{x}_i = \boldsymbol{g}_{i,y_i,r}$.

5.2 Many-Constraints Updates

A more accurate approximation can be obtained by selecting multiple constraints. Analogously, we choose the $k \leq K-1$ constraints corresponding to the labels $r_1, \ldots, r_k \neq y_i$ that achieve the highest predicted ranks. The resulting optimization is convex and can be solved by a standard Hildreth-like algorithm (Censor & Zenios, 1997). We refer to this update as `Exact`. However, `Exact` is expensive to compute, and tends to over-fit in practice (Sec. 6.2). We propose several approximate alternatives.

Sequential Update: The Hildreth algorithm iterates over the constraints, updating with respect to each until convergence is reached. We approximate this solution by making only a single pass:

- Set $\boldsymbol{\mu}_{i,0} = \boldsymbol{\mu}_i$ and $\Sigma_{i,0} = \Sigma_i$.
- For $j = 1, \ldots, k$, set $(\boldsymbol{\mu}_{i,j}, \Sigma_{i,j})$ to the solution of the following optimization:

$$\min_{\boldsymbol{\mu},\Sigma} \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \,\|\, \mathcal{N}\left(\boldsymbol{\mu}_{i,j-1}, \Sigma_{i,j-1}\right)\right)$$
$$\text{s.t. } \Pr\left[A_{y_i,r_j}(\boldsymbol{x})\right] \geq \eta$$

- Set $\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_{i,k}$ and $\Sigma_{i+1} = \Sigma_{i,k}$.

Parallel Update: As an alternative to the Hildreth algorithm, we consider the simultaneous algorithm of Iusem and Pierro (1987), which finds an exact solution by iterating over the constraints in parallel. As above, we approximate the exact solution by performing only one iteration. The process is as follows.

- For $j = 1, \ldots, k$, set $(\boldsymbol{\mu}_{i,j}, \Sigma_{i,j})$ to the solution of the following optimization:

$$\min_{\boldsymbol{\mu},\Sigma} \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \,\|\, \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)\right)$$
$$\text{s.t. } \Pr\left[A_{y_i,r_j}(\boldsymbol{x})\right] \geq \eta$$

- Let $\boldsymbol{\lambda}$ be a vector, $\lambda_j \geq 0$, $\sum_j \lambda_j = 1$.
- Set $\boldsymbol{\mu}_{i+1} = \sum_j \lambda_j \boldsymbol{\mu}_{i,j}$, $\Sigma_{i+1}^{-1} = \sum_j \lambda_j \Sigma_{i,j}^{-1}$.

In practice we set $\lambda_j = 1/k$ for all j.

6 Experiments

6.1 Datasets

Following the approach of Dredze et al. (2008), we evaluate using five natural language classification tasks over nine datasets that vary in difficulty, size, and label/feature counts. See Table 1 for an overview. Brief descriptions follow.

Task	*Instances*	*Features*	*Labels*	*Bal.*
20 News	18,828	252,115	20	Y
Amazon 7	13,580	686,724	7	Y
Amazon 3	7,000	494,481	3	Y
Enron A	3,000	13,559	10	N
Enron B	3,000	18,065	10	N
NYTD	10,000	108,671	26	N
NYTO	10,000	108,671	34	N
NYTS	10,000	114,316	20	N
Reuters	4,000	23,699	4	N

Table 1: A summary of the nine datasets, including the number of instances, features, and labels, and whether the numbers of examples in each class are balanced.

Task	*Sequential*	*Parallel*	*Exact*
20 News	**92.16**	91.41	88.08
Amazon 7	77.98	**78.35**	77.92
Amazon 3	93.54	**93.81**	93.00
Enron A	**82.40**	81.30	77.07
Enron B	71.80	**72.13**	68.00
NYTD	**83.43**	81.43	80.92
NYTO	**82.02**	78.67	80.60
NYTS	52.96	**54.78**	51.62
Reuters	93.60	**93.97**	93.47

Table 2: A comparison of $k = \infty$ updates. While the two approximations (sequential and parallel) are roughly the same, the exact solution over-fits.

Amazon Amazon product reviews. Using the data of Dredze et al. (2008), we created two domain classification datasets from seven product types (apparel, books, dvds, electronics, kitchen, music, video). *Amazon 7* includes all seven product types and *Amazon 3* includes books, dvds, and music. Feature extraction follows Blitzer et al. (2007) (bigram features and counts).

20 Newsgroups Approximately 20,000 newsgroup messages, partitioned across 20 different newsgroups.[1] This dataset is a popular choice for binary and multi-class text classification as well as unsupervised clustering. We represent each message as a binary bag-of-words.

Enron Automatic sorting of emails into folders.[2] We selected two users with many email folders and messages: `farmer-d` (*Enron A*) and `kaminski-v` (*Enron B*). We used the ten largest folders for each user, excluding non-archival email folders such as "inbox," "deleted items," and "discussion threads." Emails were represented as binary bags-of-words with stop-words removed.

NY Times To the best of our knowledge we are the first to evaluate machine learning methods on the New York Times corpus. The corpus contains 1.8 million articles that appeared from 1987 to 2007 (Sandhaus, 2008). In addition to being one of the largest collections of raw news text, it is possibly the largest collection of publicly released annotated news text, and therefore an ideal corpus for large scale NLP tasks. Among other annotations, each article is labeled with the desk that produced the story (Financial, Sports, etc.) (*NYTD*), the online section to which the article was posted (*NYTO*), and the section in which the article was printed (*NYTS*). Articles were represented as bags-of-words with feature counts (stop-words removed).

Reuters Over 800,000 manually categorized newswire stories (RCV1-v2/ LYRL2004). Each article contains one or more labels describing its general topic, industry, and region. We performed topic classification with the four general topics: corporate, economic, government, and markets. Details on document preparation and feature extraction are given by Lewis et al. (2004).

6.2 Evaluations

We first set out to compare the three update approaches proposed in Sec. 5.2: an exact solution and two approximations (sequential and parallel). Results (Table 2) show that the two approximations perform similarly. For every experiment the CW parameter η and the number of iterations (up to 10) were optimized using a single randomized iteration. However, sequential converges faster, needing an average of 4.33 iterations compared to 7.56 for parallel across all datasets. Therefore, we select sequential for our subsequent experiments.

The exact method performs poorly, displaying the lowest performance on almost every dataset. This is unsurprising given similar results for binary CW learning Dredze et al. (2008), where exact updates were shown to over-fit but converged after a single iteration of training. Similarly, our exact implementation converges after an average of 1.25 iterations, much faster than either of the approximations. However, this rapid convergence appears to come at the expense of accuracy. Fig. 1 shows the accuracy on Amazon 7 test data after each training iteration. While both sequential and parallel improve with several iterations, exact de-

[1] `http://people.csail.mit.edu/jrennie/20Newsgroups/`
[2] `http://www.cs.cmu.edu/~enron/`

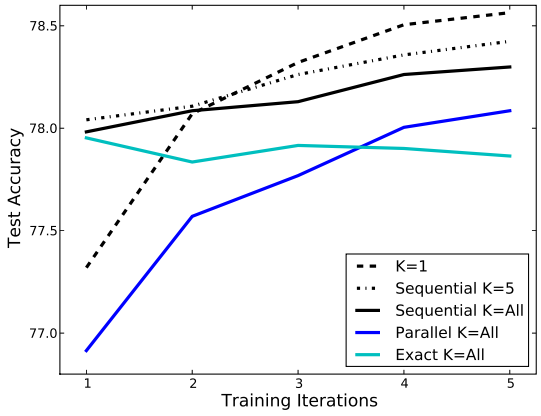


Figure 1: Accuracy on test data after each iteration on the Amazon 7 dataset.

grades after the first iteration, suggesting that it may over-fit to the training data. The approximations appear to smooth learning and produce better performance in the long run.

6.3 Relaxing Many-Constraints

While enforcing many constraints may seem optimal, there are advantages to pruning the constraints as well. It may be time consuming to enforce dozens or hundreds of constraints for tasks with many labels. Structured prediction tasks often involve exponentially many constraints, making pruning mandatory. Furthermore, many real world datasets, especially in NLP, are noisy, and enforcing too many constraints can lead to over-fitting. Therefore, we consider the impact of reducing the constraint set in terms of both reducing run-time and improving accuracy.

We compared using all constraints ($k = \infty$) with using 5 constraints ($k = 5$) for the sequential update method (Table 3). First, we observe that $k = 5$ performs better than $k = \infty$ on nearly every dataset: fewer constraints help avoid over-fitting and once again, simpler is better. Additionally, $k = 5$ converges faster than $k = \infty$ in an average of 2.22 iterations compared with 4.33 iterations. Therefore, reducing the number of constraints improves both speed and accuracy. In comparing $k = 5$ with the further reduced $k = 1$ results, we observe the latter improves on seven of the nine methods. This surprising result suggests that CW learning can perform well even without considering more than a single constraint per example. However, $k = 1$ exceeds the performance of multiple constraints only through repeated training iterations. $k = 5$ CW learning converges faster — 2.22 iterations compared with 6.67 for $k = 1$ — a desirable property in many resource restricted settings. (In the true online setting, only a single iteration may be possible.) Fig. 1 plots the performance of $k = 1$ and $k = 5$ CW on test data after each training iteration. While $k = 1$ does better in the long run, it lags behind $k = 5$ for several iterations. In fact, after a single training iteration, $k = 5$ outperforms $k = 1$ on eight out of nine datasets. Thus, there is again a tradeoff between faster convergence ($k = 5$) and increased accuracy ($k = 1$). While the $k = 5$ update takes longer per iteration, the time required for the approximate solutions grows only linearly in the number of constraints. The evaluation in Fig. 1 required 3 seconds for the first iteration of $k = 1$, 10 seconds for $k = 5$ and 11 seconds for one iteration of all 7 constraints. These differences are insignificant compared to the cost of performing multiple iterations over a large dataset. We note that, while both approximate methods took about the same amount of time, the exact solution took over 4 minutes for its first iteration.

Finally, we compare CW methods with several baselines in Table 3. Online baselines include Top-1 Perceptron (Collins, 2002), Top-1 Passive-Aggressive (PA), and k-best PA (Crammer & Singer, 2003; McDonald et al., 2004). Batch algorithms include Maximum Entropy (default configuration in McCallum (2002)) and support vector machines (LibSVM (Chang & Lin, 2001) for one-against-one classification and multiclass (MC) (Crammer & Singer, 2001)). Classifier parameters (C for PA/SVM and maxent's Gaussian prior) and number of iterations (up to 10) for the online methods were optimized using a single randomized iteration. On eight of the nine datasets, CW improves over all baselines. In general, CW provides faster and more accurate multiclass predictions.

7 Error and Probabilistic Output

Our focus so far has been on accuracy and speed. However, there are other important considerations for selecting learning algorithms. Maximum entropy and other probabilistic classification algorithms are sometimes favored for their probability scores, which can be useful for integration with other learning systems. However, practition-

		PA		CW			SVM		
Task	*Perceptron*	*K=1*	*K=5*	*K=1*	*K=5*	*K=∞*	*1 vs. 1*	*MC*	*Maxent*
20 News	81.07	88.59	88.60	$**$**92.90**	$**$92.78	$**$92.16	85.18	90.33	88.94
Amazon 7	74.93	76.55	76.72	$**$**78.70**	$**$78.04	$**$77.98	75.11	76.60	76.40
Amazon 3	92.26	92.47	93.29	†94.01	$**$**94.29**	93.54	92.83	93.60	93.60
Enron A	74.23	79.27	80.77	††**83.83**	†82.23	†82.40	80.23	82.60	82.80
Enron B	66.30	69.93	68.90	$**$**73.57**	$**$72.27	$**$71.80	65.97	71.87	69.47
NYTD	80.67	83.12	81.31	$**$**84.57**	$*$83.94	83.43	82.95	82.00	83.54
NYTO	78.47	81.93	81.22	†**82.72**	†82.55	82.02	82.13	81.01	82.53
NYTS	50.80	**56.19**	55.04	54.67	54.26	52.96	55.81	56.74	53.82
Reuters	92.10	93.12	93.30	93.60	**93.67**	93.60	92.97	93.32	93.40

Table 3: A comparison of CW learning ($k = 1, 5, \infty$ with sequential updates) with several baseline algorithms. CW learning achieves the best performance eight out of nine times. Statistical significance (McNemar) is measured against *all* baselines ($*$ indicates 0.05 and $**$ 0.001) or against online baselines († indicates 0.05 and †† 0.001).

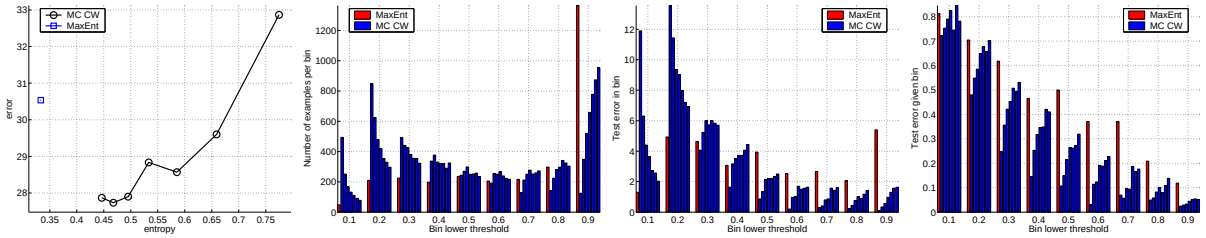


Figure 2: **First panel:** Error versus prediction entropy on Enron B. As CW converges (right to left) error and entropy are reduced. **Second panel:** Number of test examples per prediction probability bin. The red bars correspond to maxent and the blue bars to CW, with increasing numbers of epochs from left to right. **Third panel:** The contribution of each bin to the total test error. **Fourth panel:** Test error conditioned on prediction probability.

ers have observed that maxent probabilities can have low entropy and be unreliable for estimating prediction confidence (Malkin & Bilmes, 2008). Since CW also produces label probabilities — and does so in a conceptually distinct way — we investigate in this section some empirical properties of the label distributions induced by CW's parameter distributions and compare them with those of maxent.

We trained maxent and CW $k = 1$ classifiers on the Enron B dataset, optimizing parameters as before (maxent's Gaussian prior and CW's η). We estimated the label distributions from our CW classifiers after each iteration and on every test example $\boldsymbol{x}$ by Gibbs sampling weight vectors $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, and for each label y counting the fraction of weight vectors for which $y = \arg\max_z \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, z)$. Normalizing these counts yields the label distributions $\Pr[y|\boldsymbol{x}]$. We denote by $\hat{y}$ the predicted label for a given $\boldsymbol{x}$, and refer to $\Pr[\hat{y}|\boldsymbol{x}]$ as the prediction probability.

The leftmost panel of Fig. 2 plots each method's prediction error against the normalized entropy of the label distribution $-\left(\frac{1}{m}\sum_i \sum_z \Pr[z|\boldsymbol{x}_i] \log\left(\Pr[z|\boldsymbol{x}_i]\right)\right) / \log(K)$. Each CW iteration (moving from right to left in the plot) reduces both error and entropy. From our maxent results we make the common observation that maxent distributions have (ironically) low entropy. In contrast, while CW accuracy exceeds maxent after its second iteration, normalized entropy remains high. Higher entropy suggests a distribution over labels that is less peaked and potentially more informative than those from maxent. We found that the average probability assigned to a correct prediction was 0.75 for CW versus 0.83 for maxent and for an incorrect prediction was 0.44 for CW versus 0.56 for maxent.

Next, we investigate how these probabilities relate to label accuracy. In the remaining panels, we binned examples according to their prediction probabilities $\Pr[\hat{y}|\boldsymbol{x}] = \max_y \Pr[y|\boldsymbol{x}]$. The second panel of Fig. 2 shows the numbers of test examples with $\Pr[\hat{y}|\boldsymbol{x}] \in [\theta, \theta + 0.1)$ for $\theta = 0.1, 0.2 \ldots 0.9$. (Note that since there are 10

classes in this problem, we must have $\Pr[\hat{y}|\boldsymbol{x}] \geq 0.1$.) The red (leftmost) bar corresponds to the maximum entropy classifier, and the blue bars correspond, from left to right, to CW after each successive training epoch.

From the plot we observe that the maxent classifier assigns prediction probability greater than 0.9 to more than 1,200 test examples out of 3,000. Only 50 examples predicted by maxent fall in the lowest bin, and the rest of examples are distributed nearly uniformly across the remaining bins. The large number of examples with very high prediction probability explains the low entropy observed for the maximum entropy classifier.

In contrast, the CW classifier shows the opposite behavior after one epoch of training (the leftmost blue bar), assigning low prediction probability (less than 0.3) to more than 1,200 examples and prediction probability of at least 0.9 to only 100 examples. As CW makes additional passes over the training data, its prediction confidence increases and shifts toward more peaked distributions. After seven epochs fewer than 100 examples have low prediction probability and almost 1,000 have high prediction probability. Nonetheless, we note that this distribution is still less skewed than that of the maximum entropy classifier.

Given the frequency of high probability maxent predictions, it seems likely that many of the high probability maxent labels will be wrong. This is demonstrated in the third panel, which shows the contribution of each bin to the total test error. Each bar reflects the number of mistakes per bin divided by the size of the complete test set (3,000). Thus, the sum of the heights of the corresponding bars in each bin is proportional to test error. Much of the error of the maxent classifier comes not only from the low-probability bins, due to their inaccuracy, but also from the highest bin, due to its very high population. In contrast, the CW classifiers see very little error contribution from the high-probability bins. As training progresses, we see again that the CW classifiers move in the direction of the maxent classifier but remain essentially unimodal.

Finally, the rightmost panel shows the conditional test error given bin identity, or the fraction of test examples from each bin where the prediction was incorrect. This is the pointwise ratio between corresponding values of the previous two histograms. For both methods, there is a monotonically decreasing trend in error as prediction probability increases; that is, the higher the value of the prediction probability, the more likely that the prediction it provides is correct. As CW is trained, we see an increase in the conditional test error, yet the overall error decreases (not shown). This suggests that as CW is trained and its overall accuracy improves, there are more examples with high prediction probability, and the cost for this is a relative increase in the conditional test error per bin. The maxent classifier produces an extremely large number of test examples with very high prediction probabilities, which yields relatively high conditional test error. In nearly all cases, the conditional error values for the CW classifiers are smaller than the corresponding values for maximum entropy. These observations suggest that CW assigns probabilities more conservatively than maxent does, and that the (fewer) high confidence predictions it makes are of a higher quality. This is a potentially valuable property, e.g., for system combination.

8 Conclusion

We have proposed a series of approximations for multi-class confidence weighted learning, where the simple analytical solutions of binary CW learning do not apply. Our best CW method outperforms online and batch baselines on eight of nine NLP tasks, and is highly scalable due to the use of a single optimization constraint. Alternatively, our multi-constraint algorithms provide improved performance for systems that can afford only a single pass through the training data, as in the true online setting. This result stands in contrast to previously observed behaviors in non-CW settings (McDonald et al., 2004). Additionally, we found improvements in both label entropy and accuracy as compared to a maximum entropy classifier. We plan to extend these ideas to structured problems with exponentially many labels and develop methods that efficiently model label correlations. An implementation of CW multi-class algorithms is available upon request from the authors.

References

Blitzer, J., Dredze, M., & Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *Association for Computational Linguistics (ACL)*.

Censor, Y., & Zenios, S. (1997). *Parallel optimization: Theory, algorithms, and applications*. Oxford University Press, New York, NY, USA.

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

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. *Empirical Methods in Natural Language Processing (EMNLP)*.

Crammer, K., Dredze, M., & Pereira, F. (2008). Exact confidence-weighted learning. *Advances in Neural Information Processing Systems 22*.

Crammer, K., & Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Jornal of Machine Learning Research*, *2*, 265–292.

Crammer, K., & Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *Jornal of Machine Learning Research (JMLR)*, *3*, 951–991.

Dredze, M., Crammer, K., & Pereira, F. (2008). Confidence-weighted linear classification. *International Conference on Machine Learning (ICML)*.

Iusem, A., & Pierro, A. D. (1987). A simultaneous iterative method for computing projections on polyhedra. *SIAM J. Control and Optimization*, *25*.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research (JMLR)*, *5*, 361–397.

Malkin, J., & Bilmes, J. (2008). Ratio semi-definite classifiers. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*.

McCallum, A. (2002). MALLET: A machine learning for language toolkit. `http://mallet.cs.umass.edu`.

McDonald, R., Crammer, K., & Pereira, F. (2004). Large margin online learning algorithms for scalable structured classification. *NIPS Workshop on Structured Outputs*.

Sandhaus, E. (2008). The new york times annotated corpus. Linguistic Data Consortium, Philadelphia.

Model Adaptation via Model Interpolation and Boosting for Web Search Ranking

Jianfeng Gao[*], Qiang Wu[*], Chris Burges[*], Krysta Svore[*],
Yi Su[#], Nazan Khan[$], Shalin Shah[$], Hongyan Zhou[$]
[*]Microsoft Research, Redmond, USA
{jfgao; qiangwu; cburges; ksvore}@microsoft.com
[#]Johns Hopkins University, USA
suy@jhu.edu
[$]Microsoft Bing Search, Redmond, USA
{nazanka; a-shas; honzhou}@microsoft.com

Abstract

This paper explores two classes of model adaptation methods for Web search ranking: Model Interpolation and error-driven learning approaches based on a boosting algorithm. The results show that model interpolation, though simple, achieves the best results on all the open test sets where the test data is very different from the training data. The tree-based boosting algorithm achieves the best performance on most of the closed test sets where the test data and the training data are similar, but its performance drops significantly on the open test sets due to the instability of trees. Several methods are explored to improve the robustness of the algorithm, with limited success.

1 Introduction

We consider the task of ranking Web search results, i.e., a set of retrieved Web documents (URLs) are ordered by relevance to a query issued by a user. In this paper we assume that the task is performed using a *ranking model* (also called *ranker* for short) that is learned on labeled training data (e.g., human-judged query-document pairs). The ranking model acts as a function that maps the feature vector of a query-document pair to a real-valued score of relevance.

Recent research shows that such a learned ranker is superior to classical retrieval models in two aspects (Burges et al., 2005; 2006; Gao et al., 2005). First, the ranking model can use arbitrary features. Both traditional criteria such as TF-IDF and BM25, and non-traditional features such as hyperlinks can be incorporated as features in the ranker. Second, if large amounts of high-quality human-judged query-document pairs were available for model training, the ranker could achieve significantly better retrieval results than the traditional retrieval models that cannot benefit from training data effectively. However, such training data is not always available for many search domains, such as non-English search markets or person name search.

One of the most widely used strategies to remedy this problem is model adaptation, which attempts to adjust the parameters and/or structure of a model trained on one domain (called the *background* domain), for which large amounts of training data are available, to a different domain (the *adaptation* domain), for which only small amounts of training data are available. In Web search applications, domains can be defined by query types (e.g., person name queries), or languages, etc.

In this paper we investigate two classes of model adaptation methods for Web search ranking: Model Interpolation approaches and error-driven learning approaches. In model interpolation approaches, the adaptation data is used to derive a domain-specific model (also called in-domain model), which is then combined with the background model trained on the background data. This appealingly simple concept provides fertile ground for experimentation, depending on the level at which the combination is implemented (Bellegarda, 2004). In error-driven learning approaches, the background model is adjusted so as to minimize the ranking errors the model makes on the adaptation data (Bacchiani et al., 2004; Gao et al. 2006). This is arguably more powerful than model interpolation for two reasons. First, by defining a proper error function, the method can optimize more directly the measure used to assess the final quality of the Web search system, e.g., *Normalized Discounted Cumulative Gain* (Javelin & Kekalainen, 2000) in this study. Second, in this framework, the model can be adjusted to be as fine-grained as necessary. In this study we developed a set of error-driven learning methods based on a boosting algorithm where, in an incremental manner, not only each feature weight could be

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 505–513,
Singapore, 6-7 August 2009.

changed separately, but new features could be constructed.

We focus our experiments on the robustness of the adaptation methods. A model is robust if it performs reasonably well on unseen test data that could be significantly different from training data. Robustness is important in Web search applications. Labeling training data takes time. As a result of the dynamic nature of Web, by the time the ranker is trained and deployed, the training data may be more or less out of date. Our results show that the model interpolation is much more robust than the boosting-based methods. We then explore several methods to improve the robustness of the methods, including regularization, randomization, and using shallow trees, with limited success.

2 Ranking Model and Quality Measure in Web Search

This section reviews briefly a particular example of rankers, called LambdaRank (Burges et al., 2006), which serves as the baseline ranker in our study.

Assume that training data is a set of input/output pairs ($\mathbf{x}$, y). $\mathbf{x}$ is a feature vector extracted from a query-document pair. We use approximately 400 features, including dynamic ranking features such as term frequency and BM25, and statistic ranking features such as PageRank. y is a human-judged relevance score, 0 to 4, with 4 as the most relevant.

LambdaRank is a neural net ranker that maps a feature vector $\mathbf{x}$ to a real value y that indicates the relevance of the document given the query (relevance score). For example, a linear LambdaRank simply maps $\mathbf{x}$ to y with a learned weight vector $\mathbf{w}$ such that $y = \mathbf{w} \cdot \mathbf{x}$. (We used nonlinear LambdaRank in our experiments). LambdaRank is particularly interesting to us due to the way $\mathbf{w}$ is learned. Typically, $\mathbf{w}$ is optimized w.r.t. a cost function using numerical methods if the cost function is smooth and its gradient w.r.t. $\mathbf{w}$ can be computed easily. In order for the ranker to achieve the best performance in document retrieval, the cost function used in training should be the same as, or as close as possible to, the measure used to assess the quality of the system. In Web search, *Normalized Discounted Cumulative Gain* (NDCG) (Jarvelin and Kekalainen, 2000) is widely used as quality measure. For a query, NDCG is computed as

$$\mathcal{N}_i = N_i \sum_{j=1}^{L} \frac{2^{r(j)} - 1}{\log(1+j)}, \qquad (1)$$

where $r(j)$ is the relevance level of the j-th document, and the normalization constant N_i is chosen so that a perfect ordering would result in $\mathcal{N}_i = 1$. Here L is the ranking truncation level at which NDCG is computed. The $\mathcal{N}_i$ are then averaged over a query set. However, NDCG, if it were to be used as a cost function, is either flat or discontinuous everywhere, and thus presents challenges to most optimization approaches that require the computation of the gradient of the cost function.

LambdaRank solves the problem by using an implicit cost function whose gradients are specified by rules. These rules are called *λ-functions*. Burges et al. (2006) studied several λ-functions that were designed with the NDCG cost function in mind. They showed that LambdaRank with the best λ-function outperforms significantly a similar neural net ranker, RankNet (Burges et al., 2005), whose parameters are optimized using the cost function based on cross-entropy.

The superiority of LambdaRank illustrates the key idea based on which we develop the model adaptation methods. We should always adapt the ranking models in such a way that the NDCG can be optimized as directly as possible.

3 Model Interpolation

One of the simplest model interpolation methods is to combine an in-domain model with a background model at the model level via linear interpolation. In practice we could combine more than two in-domain/background models. Letting *Score*(q, d) be a ranking model that maps a query-document pair to a relevance score, the general form of the interpolation model is

$$Score(q,d) = \sum_{i=1}^{N} \alpha_i Score_i(q,d), \qquad (2)$$

where the α's are interpolation weights, optimized on validation data with respect to a predefined objective, which is NDCG in our case. As mentioned in Section 2, NDCG is not easy to optimize, for which we resort to two solutions, both of which achieve similar results in our experiments.

The first solution is to view the interpolation model of Equation (2) as a linear neural net ranker where each component model $Score_i(.)$ is defined as a feature function. Then, we can use the LambdaRank algorithm described in Section 2 to find the optimal weights.

An alternative solution is to view interpolation weight estimation as a multi-dimensional optimization problem, with each model as a

dimension. Since NCDG is not differentiable, we tried in our experiments the numerical algorithms that do not require the computation of gradient. Among the best performers is the Powell Search algorithm (Press et al., 1992). It first constructs a set of N virtual directions that are conjugate (i.e., independent with each other), then it uses *line search* N times, each on one virtual direction, to find the optimum. Line search is a one-dimensional optimization algorithm. Our implementation follows the one described in Gao et al. (2005), which is used to optimize the averaged precision.

The performance of model interpolation depends to a large degree upon the quality and the size of adaptation data. First of all, the adaptation data has to be "rich" enough to suitably characterize the new domain. This can only be achieved by collecting more in-domain data. Second, once the domain has been characterized, the adaptation data has to be "large" enough to have a model reliably trained. For this, we developed a method, which attempts to augment adaptation data by gathering similar data from background data sets.

The method is based on the *k-nearest-neighbor* (kNN) algorithm, and is inspired by Bishop (1995). We use the small in-domain data set *D1* as a seed, and expand it using the large background data set *D2*. When the relevance labels are assigned by humans, it is reasonable to assume that queries with the lowest information entropy of labels are the least noisy. That is, for such a query most of the URLs are labeled as highly relevant/not relevant documents rather than as moderately relevance/not relevant documents.

Due to computational limitations of kNN-based algorithms, a small subset of queries from *D1* which are least noisy are selected. This data set is called *S1*. For each sample in *D2*, its 3-nearest neighbors in *S1* are found using a cosine-similarity metric. If the three neighbors are within a very small distance from the sample in *D2*, and one of the labels of the nearest neighbors matches exactly, the training sample is selected and is added to the expanded set *E2*, in its own query. This way, *S1* is used to choose training data from *D2*, which are found to be close in some space.

This process effectively creates several data points in close neighborhood of the points in the original small data set *D1*, thus expanding the set, by jittering each training sample a little. This is equivalent to training with noise (Bishop, 1995), except that the training samples used are actual queries judged by a human. This is found to increase the NDCG in our experiments.

1 Set $F_0(\mathbf{x})$ be the background ranking model
2 **for** $m = 1$ **to** M **do**
3 $\quad y_i' = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$, for $i = 1 \ldots N$
4 $\quad (h_m, \beta_m) = \underset{h,\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left[y_i' - \beta h(\mathbf{x}_i)\right]^2$
5 $\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m h(\mathbf{x})$

Figure 1. The generic boosting algorithm for model adaptation

4 Error-Driven Learning

Our error-drive learning approaches to ranking modeling adaptation are based on the Stochastic Gradient Boosting algorithm (or the boosting algorithm for short) described in Friedman (1999). Below, we follow the notations in Friedman (2001).

Let adaptation data (also called *training data* in this section) be a set of input/output pairs $\{\mathbf{x}_i, y_i\}$, $i = 1 \ldots N$. In error-driven learning approaches, model adaptation is performed by adjusting the background model into a new in-domain model $F: x \rightarrow y$ that minimizes a loss function $L(y, F(\mathbf{x}))$ over all samples in training data

$$F^* = \underset{F}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, F(\mathbf{x}_i)). \quad (3)$$

We further assume that $F(\mathbf{x})$ takes the form of additive expansion as

$$F(\mathbf{x}) = \sum_{m=0}^{M} \beta_m h(\mathbf{x}; \mathbf{a}_m), \quad (4)$$

where $h(\mathbf{x}; \mathbf{a})$ is called *basis function*, and is usually a simple parameterized function of the input $\mathbf{x}$, characterized by parameters $\mathbf{a}$. In what follows, we drop $\mathbf{a}$, and use $h(\mathbf{x})$ for short. In practice, the form of h has to be restricted to a specific function family to allow for a practically efficient procedure of model adaptation. β is a real-valued coefficient.

Figure 1 is the generic algorithm. It starts with a base model F_0, which is a background model. Then for $m = 1, 2, \ldots, M$, the algorithm takes three steps to adapt the base model so as to best fit the adaptation data: (1) compute the residual of the current base model (line 3), (2) select the optimal basis function (line 4) that best fits the residual, and (3) update the base model by adding the optimal basis function (line 5). The two model adaptation algorithms that will be described below follow the same 3-step adaptation procedure. They only differ in the choice of h. In the LambdaBoost algorithm (Section 4.1) h

1 Set $F_0(\mathbf{x})$ to be the background ranking model
2 **for** $m = 1$ **to** M **do**
3 compute residuals according to Equation (5)
4 select best h_m (with its best β_m), according to *LS*, computed by Equations (8) and (9)
5 $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \upsilon\beta_m h(\mathbf{x})$

Figure 2. The LambdaBoost algorithm for model adaptation.

is defined as a single feature, and in LambdaSMART (Section 4.2), h is a regression tree.

Now, we describe the way residual is computed, the step that is identical in both algorithms. Intuitively, the residual, denoted by y' (line 3 in Figure 1), measures the amount of errors (or loss) the base model makes on the training samples. If the loss function in Equation (3) is differentiable, the residual can be computed easily as the negative gradient of the loss function. As discussed in Section 2, we want to directly optimize the NDCD, whose gradient is approximated via the λ-function. Following Burges et al. (2006), the gradient of a training sample $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is a feature vector representing the query-document pair (q_i, d_i), w.r.t. the current base model is computed by marginalizing the λ-functions of all document pairs, (d_i, d_j), of the query, q_i, as

$$y_i' = \sum_{j \neq i} \Delta\text{NDCG} \cdot \frac{\partial C_{ij}}{\partial o_{ij}}, \quad (5)$$

where ΔNDCG is the NDCG gained by swapping those two documents (after sorting all documents by their current scores); $o_{ij} \equiv s_i - s_j$ is the difference in ranking scores of d_i and d_j given q_i; and C_{ij} is the cross entropy cost defined as

$$C_{ij} \equiv C(o_{ij}) = s_j - s_i + \log(1 + \exp(s_i - s_j)). \quad (6)$$

Thus, we have

$$\frac{\partial C_{ij}}{\partial o_{ij}} = \frac{-1}{1 + \exp(o_{ij})}. \quad (7)$$

This λ-function essentially uses the cross entropy cost to smooth the change in NDCG obtained by swapping the two documents. A key intuition behind the λ-function is the observation that NDCG does not treat all pairs equally; for example, it costs more to incorrectly order a pair, where the irrelevant document is ranked higher than a highly relevant document, than it does to swap a moderately relevant/not relevant pair.

4.1 The LambdaBoost Algorithm

In LambdaBoost, the basis function h is defined as a single feature (i.e., an element feature in the feature vector $\mathbf{x}$). The algorithm is summarized in Figure 2. It iteratively adapts a background model to training data using the 3-step procedure, as in Figure 1. Step 1 (line 3 in Figure 2) has been described.

Step 2 (line 4 in Figure 2) finds the optimal basis function h, as well as its optimal coefficient β, that best fits the residual according to the least-squares (LS) criterion. Formally, let h and β denote the candidate basis function and its optimal coefficient. The LS error on training data is $LS(h;\beta) = \sum_{i=0}^{N}(y_i' - \beta h)^2$, where y_i' is computed as Equation (5). The optimal coefficient of h is estimated by solving the equation $\partial \sum_{i=1}^{N}(y_i' - \beta h2/\partial\beta=0$. Then, β is computed as

$$\beta = \frac{\sum_{i=1}^{N} y_i' h(\mathbf{x}_i)}{\sum_{i=1}^{N} h(\mathbf{x}_i)}. \quad (8)$$

Finally, given its optimal coefficient β, the optimal LS loss of h is

$$LS(h;\beta) = \sum_{i=1}^{N} y_i' \times y_i' - \frac{(\sum_{i=1}^{N} y_i' h(\mathbf{x}_i))^2}{\sum_{i=1}^{N} h^2(\mathbf{x}_i)}. \quad (9)$$

Step 3 (line 5 in Figure 2) updates the base model by adding the chosen optimal basis function with its optimal coefficient. As shown in Step 2, the optimal coefficient of each candidate basis function is computed when the basis function is evaluated. However, adding the basis function using its optimal efficient is prone to overfitting. We thus add a shrinkage coefficient $0 < \upsilon < 1$ – the fraction of the optimal line step taken. The update equation is thus rewritten in line 5 in Figure 2.

Notice that if the background model contains all the input features in $\mathbf{x}$, then LambdaBoost does not add any new features but adjust the weights of existing features. If the background model does not contain all of the input features, then LambdaBoost can be viewed as a feature selection method, similar to Collins (2000), where at each iteration the feature that has the largest impact on reducing training loss is selected and added to the background model. In either case, LambdaBoost adapts the background model by adding a model whose form is a (weighted) linear combination of input features. The property of linearity makes LambdaBoost robust and less likely to overfit in Web search applications. But this also limits the adaptation capacity. A simple method that allows us to go beyond linear adaptation is to define h as nonlinear terms of the input features, such as regression trees in LambdaSMART.

4.2 The LambdaSMART Algorithm

LambdaSMART was originally proposed in Wu et al. (2008). It is built on MART (Friedman, 2001) but uses the λ-function (Burges et a., 2006) to

1 Set $F_0(\mathbf{x})$ to be the background ranking model
2 **for** $m = 1$ **to** M **do**
3 compute residuals according to Equation (5)
4 create a L-terminal node tree, $h_m \equiv \{R_{lm}\}_{l=1\ldots L}$
5 **for** $l = 1$ **to** L **do**
6 compute the optimal β_{lm} according to Equation (10), based on approximate Newton step.
7 $F_m(\mathbf{x}) = F_{m-1}(x) + \upsilon \sum_{l=1\ldots L} \beta_{lm} 1(x \in R_{lm})$

Figure 3. The LambdaSMART algorithm for model adaptation.

compute gradients. The algorithm is summarized in Figure 3. Similar to LambdaBoost, it takes M rounds, and at each boosting iteration, it adapts the background model to training data using the 3-step procedure. Step 1 (line 3 in Figure 3) has been described.

Step 2 (lines 4 to 6) searches for the optimal basis function h to best fit the residual. Unlike LambdaBoost where there are a finite number of candidate basis functions, the function space of regression trees is infinite. We define h as a regression tree with L terminal nodes. In line 4, a regression tree is built using Mean Square Error to determine the best split at any node in the tree. The value associated with a leaf (i.e., terminal node) of the trained tree is computed first as the residual (computed via λ-function) for the training samples that land at that leaf. Then, since each leaf corresponds to a different mean, a one-dimensional Newton-Raphson line step is computed for each leaf (lines 5 and 6). These line steps may be simply computed as the derivatives of the LambdaRank gradients w.r.t. the model scores s_i. Formally, the value of the l-th leaf, β_{ml}, is computed as

$$\beta_{ml} = \frac{\sum_{x \in R_{lm}} y_i'}{\sum_{x \in R_{lm}} w_i}, \quad (10)$$

where y_i' is the residual of training sample i, computed in Equation (5), and w_i is the derivative of y_i', i.e., $w_i = \partial y_i' / \partial F(\mathbf{x}_i)$.

In Step 3 (line 7), the regression tree is added to the current base model, weighted by the shrinkage coefficient $0 < \upsilon < 1$.

Notice that since a regression tree can be viewed as a complex feature that combines multiple input features, LambdaSMART can be used as a feature generation method. LambdaSMART is arguably more powerful than LambdaBoost in that it introduces new complex features and thus adjusts not only the parameters but also the structure of the background model[1]. However, one problem of trees is their high variance. Often a small change in the data can result in a very different series of splits. As a result, tree-based ranking models are much less robust to noise, as we will show in our experiments. In addition to the use of shrinkage coefficient $0 < \upsilon < 1$, which is a form of model regularization according to Hastie, et al., (2001), we will explore in Section 5.3 other methods of improving the model robustness, including randomization and using shallow trees.

5 Experiments

5.1 The Data

We evaluated the ranking model adaptation methods on two Web search domains, namely (1) a name query domain, which consists of only person name queries, and (2) a Korean query domain, which consists of queries that users submitted to the Korean market.

For each domain, we used two in-domain data sets that contain queries sampled respectively from the query log of a commercial Web search engine that were collected in two non-overlapping periods of time. We used the more recent one as *open test* set, and split the other into three non-overlapping data sets, namely training, validation and *closed test* sets, respectively. This setting provides a good simulation to the realistic Web search scenario, where the rankers in use are usually trained on early collected data, and thus helps us investigate the robustness of these model adaptation methods.

The statistics of the data sets used in our person name domain adaptation experiments are shown in Table 1. The names query set serves as the adaptation domains, and Web-1 as the background domain. Since Web-1 is used to train a background ranker, we did not split it to train/valid/test sets. We used 416 input features in these experiments.

For cross-domain adaptation experiments from non-Korean to Korean markets, Korean data serves as the adaptation domain, and English, Chinese, and Japanese data sets as the background domain. Again, we did not split the data sets in the background domain to train/valid/test sets. The statistics of these data sets are shown in Table 2. We used 425 input features in these experiments.

[1] Note that in a sense our proposed LambdaBoost algorithm is the same as LambdaSMART, but using a single feature at each iteration, rather than a tree. In particular, they share the trick of using the Lambda gradients to learn NDCG.

Coll.	Description	#qry.	# url/qry
Web-1	*Background training data*	31555	134
Names-1-Train	*In-domain training data (adaptation data)*	5752	85
Names-1-Valid	*In-domain validation data*	158	154
Names-1-Test	*Closed test data*	318	153
Names-2-Test	*Open test data*	4370	84

Table 1. Data sets in the names query domain experiments, where # qry is number of queries, and # url/qry is number of documents per query.

Coll.	Description	# qry.	# url/qry
Web-En	*Background En training data*	6167	198
Web-Ja	*Background Ja training data*	45012	58
Web-Cn	*Background Ch training data*	32827	72
Kokr-1-Train	*In-domain Ko training data (adaptation data)*	3724	64
Kokr-1-Valid	*In-domain validation data*	334	130
Kokr-1-Test	*Korean closed test data*	372	126
Kokr-2-Test	*Korean open test data*	871	171

Table 2. Data sets in the Korean domain experiments.

# Models	NDCG@1	NDCG@3	NDCG@10	AveNDCG
1 Back.	0.4575	0.4952	0.5446	0.5092
2 In-domain	0.4921	0.5296	0.5774	0.5433
3 2W-Interp.	0.4745	0.5254	0.5747	0.5391
4 3W-Interp.	0.4829	0.5333	0.5814	0.5454
5 λ-Boost	0.4706	0.5011	0.5569	0.5192
6 λ-SMART	0.5042	0.5449	0.5951	0.5623

Table 3. Close test results on Names-1-Test.

# Models	NDCG@1	NDCG@3	NDCG@10	AveNDCG
1 Back.	0.5472	0.5347	0.5731	0.5510
2 In-domain	0.5216	0.5266	0.5789	0.5472
3 2W-Interp.	0.5452	0.5414	0.5891	0.5604
4 3W-Interp.	0.5474	0.5470	0.5951	0.5661
5 λ-Boost	0.5269	0.5233	0.5716	0.5428
6 λ-SMART	0.5200	0.5331	0.5875	0.5538

Table 4. Open test results on Names-2-Test.

In each domain, the in-domain training data is used to train in-domain rankers, and the background data for background rankers. Validation data is used to learn the best training parameters of the boosting algorithms, i.e., M, the total number of boosting iterations, υ, the shrinkage coefficient, and L, the number of leaf nodes for each regression tree (L=1 in LambdaBoost). Model performance is evaluated on the closed/open test sets.

All data sets contain samples labeled on a 5-level relevance scale, 0 to 4, with 4 as most relevant and 0 as irrelevant. The performance of rankers is measured through NDCG evaluated against closed/open test sets. We report NDCG scores at positions 1, 3 and 10, and the averaged NDCG score (Ave-NDCG), the arithmetic mean of the NDCG scores at 1 to 10. Significance test (i.e., t-test) was also employed.

5.2 Model Adaptation Results

This section reports the results on two adaptation experiments. The first uses a large set of Web data, Web-1, as background domain and uses the name query data set as adaptation data. The results are summarized in Tables 3 and 4. We compared the three model adaptation methods against two baselines: (1) the background ranker (Row 1 in Tables 3 and 4), a 2-layer LambdaRank model with 15 hidden nodes and a learning rate of 10^{-5} trained on Web-1; and (2) the In-domain Ranker (Row 2), a 2-layer LambdaRank model with 10 hidden nodes and a learning rate of 10^{-5} trained on Names-1-Train. We built two interpolated rankers. The 2-way interpolated ranker (Row 3) is a linear combination of the two baseline rankers, where the interpolation weights were optimized on Names-1-Valid. To build the 3-way interpolated ranker (Row 4), we linearly interpolated three rankers. In addition to the two baseline rankers, the third ranker is trained on an augmented training data, which was created using the kNN method described in Section 3.

In LambdaBoost (Row 5) and LambdaSMART (Row 6), we adapted the background ranker to name queries by boosting the background ranker with Names-1-Train. We trained LambdaBoost with the setting M = 500, υ = 0.5, optimized on Names-1-Valid. Since the background ranker uses all of the 416 input features, in each boosting iteration, LambdaBoost in fact selects one existing feature in the background ranker and adjusts its weight. We trained LambdaSMART with M = 500, L = 20, υ = 0.5, optimized on Names-1-Valid.

We see that the results on the closed test set (Table 3) are quite different from the results on the open test set (Table 4). The in-domain ranker outperforms the background ranker on the closed test set, but underperforms significantly the background ranker on the open test set. The interpretation is that the training set and the closed test set are sampled from the same data set and are very similar, but the open test set is a very different data set, as described in Section 5.1. Similarly, on the closed test set, LambdaSMART outperforms LambdaBoost with a big margin due to its superior adaptation capacity; but on the open test set their performance difference is much smaller due to the instability of the trees in LambdaSMART, as we will investigate in detail later. Interestingly, model interpolation, though simple, leads to the two best rankers on the open test set. In particular, the 3-way interpolated ranker outperforms the two baseline rankers

#	Ranker	NDCG@1	NDCG@3	NDCG@10	AveNDCG
1	Back. (En)	0.5371	0.5413	0.5873	0.5616
2	Back. (Ja)	0.5640	0.5684	0.6027	0.5808
3	Back. (Cn)	0.4966	0.5105	0.5761	0.5393
4	In-domain	0.5927	0.5824	0.6291	0.6055

Table 5. Close test results of baseline rankers, on Kokr-1-Test

#	Ranker	NDCG@1	NDCG@3	NDCG@10	AveNDCG
1	Back. (En)	0.4991	0.5242	0.5397	0.5278
2	Back. (Ja)	0.5052	0.5092	0.5377	0.5194
3	Back. (Cn)	0.4779	0.4855	0.5114	0.4942
4	In-domain	0.5164	0.5295	0.5675	0.5430

Table 6. Open test results of baseline rankers, on Kokr-2-Test

#	Ranker	NDCG@1	NDCG@3	NDCG@10	AveNDCG
1	Interp. (En)	0.5954	0.5893	0.6335	0.6088
2	Interp. (Ja)	0.6047	0.5898	0.6339	0.6116
3	Interp. (Cn)	0.5812	0.5807	0.6268	0.6024
4	4W-Interp.	0.5878	0.5870	0.6289	0.6054

Table 7. Close test results of interpolated rankers, on Kokr-1-Test.

#	Ranker	NDCG@1	NDCG@3	NDCG@10	AveNDCG
1	Interp. (En)	0.5178	0.5369	0.5768	0.5500
2	Interp. (Ja)	0.5274	0.5416	0.5788	0.5531
3	Interp. (Cn)	0.5224	0.5339	0.5766	0.5487
4	4W-Interp.	0.5278	0.5414	0.5823	0.5549

Table 8. Open test results of interpolated rankers, on Kokr-2-Test.

#	Ranker	NDCG@1	NDCG@3	NDCG@10	AveNDCG
1	λ-Boost (En)	0.5757	0.5716	0.6197	0.5935
2	λ-Boost (Ja)	0.5801	0.5807	0.6225	0.5982
3	λ-Boost (Cn)	0.5731	0.5793	0.6226	0.5972

Table 9. Close test results of λ-Boost rankers, on Kokr-1-Test.

#	Ranker	NDCG@1	NDCG@3	NDCG@10	AveNDCG
1	λ-Boost (En)	0.4960	0.5203	0.5486	0.5281
2	λ-Boost (Ja)	0.5090	0.5167	0.5374	0.5233
3	λ-Boost (Cn)	0.5177	0.5324	0.5673	0.5439

Table 10. Open test results of λ-Boost rankers, on Kokr-2-Test.

#	Ranker	NDCG@1	NDCG@3	NDCG@10	AveNDCG
1	λ-SMART (En)	0.6096	0.6057	0.6454	0.6238
2	λ- SMART (Ja)	0.6014	0.5966	0.6385	0.6172
3	λ- SMART (Cn)	0.5955	0.6095	0.6415	0.6209

Table 11. Close test results of λ-SMART rankers, on Kokr-1-Test.

#	Ranker	NDCG@1	NDCG@3	NDCG@10	AveNDCG
1	λ- SMART (En)	0.5177	0.5297	0.5563	0.5391
2	λ- SMART (Ja)	0.5205	0.5317	0.5522	0.5368
3	λ- SMART (Cn)	0.5198	0.5305	0.5644	0.5410

Table 12. Open test results of λ-SMART rankers, on Kokr-2-Test.

significantly (i.e., p-value < 0.05 according to t-test) on both the open and closed test sets.

The second adaptation experiment involves data sets from several languages (Table 2). 2-layer LambdaRank baseline rankers were first built from Korean, English, Japanese, and Chinese training data and tested on Korean test sets (Tables 5 and 6). These baseline rankers then serve as in-domain ranker and background rankers for model adaptation. For model interpolation (Tables 7 and 8), Rows 1 to 4 are three 2-way interpolated rankers built by linearly interpolating
each of the three background rankers with the in-domain ranker, respectively. Row 4 is a 4-way interpolated ranker built by interpolating the in-domain ranker with the three background rankers. For LambdaBoost (Tables 9 and 10) and LambdaSMART (Tables 11 and 12), we used the same parameter settings as those in the name query experiments, and adapted the three background rankers, to the Korean training data, Kokr-1-Train.

The results in Tables 7 to 12 confirm what we learned in the name query experiments. There are three main conclusions. (1) Model interpolation is an effective method of ranking model adaptation. E.g., the 4-way interpolated ranker outperforms other ranker significantly. (2) LambdaSMART is the best performer on the closed test set, but its performance drops significantly on the open test set due to the instability of trees. (3) LambdaBoost does not use trees. So its modeling capacity is weaker than LambdaSMART (e.g., it always underperforms LambdaSMART significantly on the closed test sets), but it is more robust due to its linearity (e.g., it performs similarly to LambdaSMART on the open test set).

5.3 Robustness of Boosting Algorithms

This section investigates the robustness issue of the boosting algorithms in more detail. We compared LambdaSMART with different values of L (i.e., the number of leaf nodes), and with and without randomization. Our assumptions are (1) allowing more leaf nodes would lead to deeper trees, and as a result, would make the resulting ranking models less robust; and (2) injecting randomness into the basis function (i.e. regression tree) estimation procedure would improve the robustness of the trained models (Breiman, 2001; Friedman, 1999). In LambdaSMART, the randomness can be injected at different levels of tree construction. We found that the most effective method is to introduce the randomness at the node level (in Step 4 in Figure 3). Before each node split, a subsample of the training data and a subsample of the features are drawn randomly. (The sample rate is 0.7). Then, the two randomly selected subsamples, instead of the full samples, are used to determine the best split.

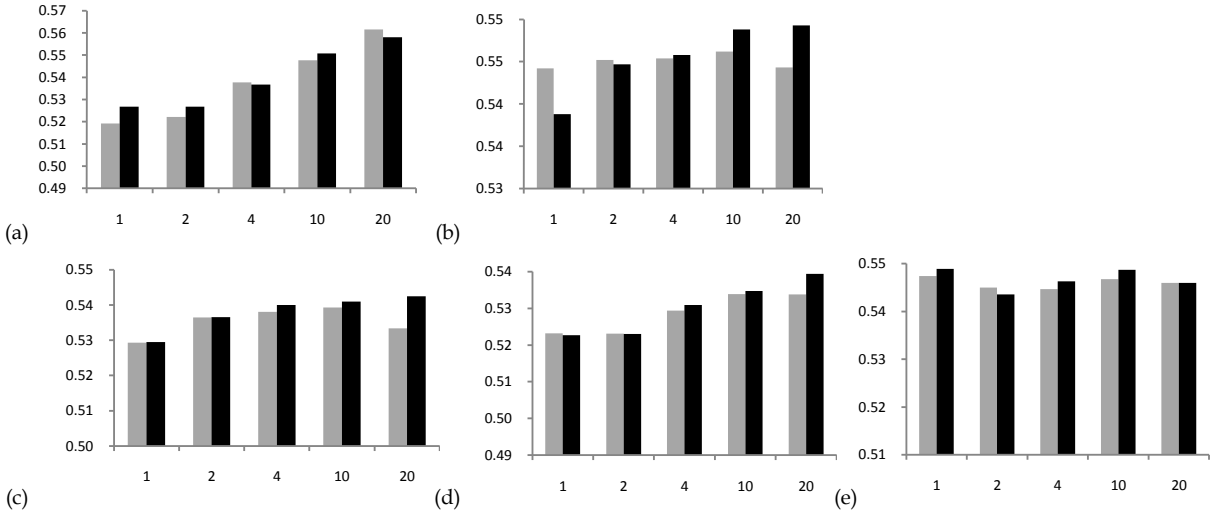


Figure 4. AveNDCG results (*y*-axis) of LambdaSMART with different values of *L* (*x*-axis), where *L*=1 is LambdaBoost; (a) and (b) are the results on closed and open tests using Names-1-Train as adaptation data, respectively; (d), (e) and (f) are the results on the Korean open test set, using background models trained on Web-En, Web-Ja, and Web-Cn data sets, respectively.

We first performed the experiments on name queries. The results on the closed and open test sets are shown in Figures 4 (a) and 4 (b), respectively. The results are consistent with our assumptions. There are three main observations. First, the gray bars in Figures 4 (a) and 4 (b) (boosting without randomization) show that on the closed test set, as expected, NDCG increases with the value of *L*, but the correlation does not hold on the open test set. Second, the black bars in these figures (boosting with randomization) show that in both closed and open test sets, NDCG increases with the value of *L*. Finally, comparing the gray bars with their corresponding black bars, we see that randomization consistently improves NDCG on the open test set, with a larger margin of gain for the boosting algorithms with deeper trees ($L > 5$).

These results are very encouraging. Randomization seems to work like a charm. Unfortunately, it does not work well enough to help the boosting algorithm beat model interpolation on the open test sets. Notice that all the LambdaSMART results reported in Section 5.2 use randomization with the same sampling rate of 0.7. We repeated the comparison in the cross-domain adaptation experiments. As shown in Figure 4, results in 4 (c) and 4 (d) are consistent with those on names queries in 4 (b). Results in 4 (f) show a visible performance drop from LambdaBoost to LambdaSMART with $L = 2$, indicating again the instability of trees.

6 Conclusions and Future Work

In this paper, we extend two classes of model adaptation methods (i.e., model interpolation and error-driven learning), which have been well studied in statistical language modeling for speech and natural language applications (e.g., Bacchiani et al., 2004; Bellegarda, 2004; Gao et al., 2006), to ranking models for Web search applications.

We have evaluated our methods on two adaptation experiments over a wide variety of datasets where the in-domain datasets bear different levels of similarities to their background datasets. We reach different conclusions from the results of the open and close tests, respectively. Our open test results show that in the cases where the in-domain data is dramatically different from the background data, model interpolation is very robust and outperforms the baseline and the error-driven learning methods significantly; whereas our close test results show that in the cases where the in-domain data is similar to the background data, the tree-based boosting algorithm (i.e. LambdaSMART) is the best performer, and achieves a significant improvement over the baselines. We also show that these different conclusions are largely due to the instability of the use of trees in the boosting algorithm. We thus explore several methods of improving the robustness of the algorithm, such as randomization, regularization, using shallow trees, with limited success. Of course, our experiments,

described in Section 5.3, only scratch the surface of what is possible. Robustness deserves more investigation and forms one area of our future work.

Another family of model adaptation methods that we have not studied in this paper is transfer learning, which has been well-studied in the machine learning community (e.g., Caruana, 1997; Marx et al., 2008). We leave it to future work.

To solve the issue of inadequate training data, in addition to model adaptation, researchers have also been exploring the use of implicit user feedback data (extracted from log files) for ranking model training (e.g., Joachims et al., 2005; Radlinski et al., 2008). Although such data is very noisy, it is of a much larger amount and is cheaper to obtain than human-labeled data. It will be interesting to apply the model adaptation methods described in this paper to adapt a ranker which is trained on a large amount of automatically extracted data to a relatively small amount of human-labeled data.

Acknowledgments

This work was done while Yi Su was visiting Microsoft Research, Redmond. We thank Steven Yao's group at Microsoft Bing Search for their help with the experiments.

References

Bacchiani, M., Roark, B. and Saraclar, M. 2004. Language model adaptation with MAP estimation and the Perceptron algorithm. In *HLT-NAACL*, 21-24.

Bellegarda, J. R. 2004. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42: 93-108.

Breiman, L. 2001. Random forests. *Machine Learning*, 45, 5-23.

Bishop, C.M. 1995. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7, 108-116.

Burges, C. J., Ragno, R., & Le, Q. V. 2006. Learning to rank with nonsmooth cost functions. In *ICML.*

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, and Hullender, G. 2005. Learning to rank using gradient descent. In *ICML.*

Caruana, R. 1997. Multitask learning. *Machine Learning*, 28(1): 41-70.

Collins, M. 2000. Discriminative reranking for natural language parsing. In *ICML.*

Donmea, P., Svore, K. and Burges. 2008. On the local optimality for NDCG. *Microsoft Technical Report*, MSR-TR-2008-179.

Friedman, J. 1999. Stochastic gradient boosting. *Technical report*, Dept. Statistics, Stanford.

Friedman, J. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics,* 29(5).

Gao, J., Qin, H., Xia, X. and Nie, J-Y. 2005. Linear discriminative models for information retrieval. In *SIGIR.*

Gao, J., Suzuki, H. and Yuan, W. 2006. An empirical study on language model adaptation. *ACM Trans on Asian Language Processing*, 5(3):207-227.

Hastie, T., Tibshirani, R. and Friedman, J. 2001. *The elements of statistical learning*. Springer-Verlag, New York.

Jarvelin, K. and Kekalainen, J. 2000. IR evaluation methods for retrieving highly relevant documents. In *SIGIR.*

Joachims, T., Granka, L., Pan, B., Hembrooke, H. and Gay, G. 2005. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR.*

Marx, Z., Rosenstein, M.T., Dietterich, T.G. and Kaelbling, L.P. 2008. Two algorithms for transfer learning. To appear in *Inductive Transfer: 10 years later*.

Press, W. H., S. A. Teukolsky, W. T. Vetterling and B. P. Flannery. 1992. *Numerical Recipes In C.* Cambridge Univ. Press.

Radlinski, F., Kurup, M. and Joachims, T. 2008. How does clickthrough data reflect retrieval quality? In *CIKM.*

Thrun, S. 1996. Is learning the *n*-th thing any easier than learning the first. In *NIPS.*

Wu, Q., Burges, C.J.C., Svore, K.M. and Gao, J. 2008. Ranking, boosting, and model adaptation. *Technical Report* MSR-TR-2008-109, Microsoft Research.

A Structural Support Vector Method for Extracting Contexts and Answers of Questions from Online Forums

Wen-Yun Yang [†*] **Yunbo Cao** [†‡] **Chin-Yew Lin** [‡]
[†] Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China
[‡] Microsoft Research Asia, Beijing, China
wenyun.yang@gmail.com {yunbo.cao; cyl}@microsoft.com

Abstract

This paper addresses the issue of extracting contexts and answers of questions from post discussion of online forums. We propose a novel and unified model by customizing the structural Support Vector Machine method. Our customization has several attractive properties: (1) it gives a comprehensive graphical representation of thread discussion. (2) It designs special inference algorithms instead of general-purpose ones. (3) It can be readily extended to different task preferences by varying loss functions. Experimental results on a real data set show that our methods are both promising and flexible.

1 Introduction

Recently, extracting questions, contexts and answers from post discussions of online forums incurs increasing academic attention (Cong et al., 2008; Ding et al., 2008). The extracted knowledge can be used either to enrich the knowledge base of community question answering (QA) services such as Yahoo! Answers or to augment the knowledge base of chatbot (Huang et al., 2007).

Figure 1 gives an example of a forum thread with questions, contexts and answers annotated. This thread contains *three posts* and *ten sentences*, among which *three questions* are discussed. The three questions are proposed in three sentences, S3, S5 and S6. The *context sentences* S1 and S2 provide contextual information for question sentence S3. Similarly, the *context sentence* S4 provides contextual information for question sentence S5 and S6. There are three question-context-answer triples in this example, $(S3) - (S1, S2) - (S8, S9)$, $(S5) - (S4) - (S10)$ and $(S6) - (S4) - (S10)$. As shown in the example, a forum question usually requires contextual information to complement its expression. For example, the question sentence S3 would be of incomplete meaning without the contexts S1 and S2, since the important keyword *pet friendly* would be lost.

Post1: <context id=1> **S1**: *Hi I am looking for a pet friendly hotel in Hong Kong because all of my family is going there for vacation.* **S2**: *my family has 2 sons and a dog.* </context> <question id=1> **S3**: *Is there any recommended hotel near Sheung Wan or Tsing Sha Tsui?* </question> <context id=2, 3> **S4**: *We also plan to go shopping in Causeway Bay.* </context> <question id=2> **S5**: *What's the traffic situation around those commercial areas?* </question> <question id=3> **S6**: *Is it necessary to take a taxi?* </question> **S7**: *Any information would be appreciated.*
Post2: <answer id=1> **S8**: *The Comfort Lodge near Kowloon Park allows pet as I know, and usually fits well within normal budgets.* **S9**: *It is also conveniently located, nearby the Kowloon railway station and subway.* </answer>
Post3: <answer id=2, 3> **S10**: *It's very crowd in those areas, so I recommend MTR in Causeway Bay because it is cheap to take you around.* </answer>

Figure 1: An example thread with three posts and ten sentences

The problem of extracting questions, contexts, and answers can be solved in two steps: (1) identify questions and then (2) extract contexts and answers for them. Since identifying questions from forum discussions is already well solved in (Cong et al., 2008), in this paper, we are focused on step (2) while assuming questions already identified.

Previously, Ding et al. (2008) employ general-purpose graphical models without any customizations to the specific extraction problem (step 2). In this paper, we improve the existing models in

[*] This work was done while the first author visited Microsoft Research Asia.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 514–523,
Singapore, 6-7 August 2009.

three aspects: *graphical representation*, *inference algorithm* and *loss function*.

Graphical representation. We propose a more comprehensive and unified graphical representation to model the thread for relational learning. Our graphical representation has two advantages over previous work (Ding et al., 2008): *unifying sentence relations* and *incorporating question interactions*.

Three types of relation should be considered for context and answer extraction: (a) relations between successive sentences (e.g., *context* sentence S2 occurs immediately before *question* sentence S3); (b) relations between *context* sentences and *answer* sentences (e.g., *context* S4 presents the phrase *Causeway Bay* linking to *answer* which is absent from *question* S6); and (c) relations between multiple labels for one sentence (e.g., one *question* sentence is unlikely to be the *answer* to another *question* although one sentence can serve as *contexts* for more than one *questions*). Our proposed graphical representation improves the modeling of the three types of sentence relation (Section 2.2).

Certain interactions exist among questions. For example, *question* sentences S5 and S6 interact by sharing *context* sentence S4. Our proposed graphical representation can naturally model the interactions. Previous work (Ding et al., 2008) performs the extraction of contexts and answers in multiple passes of the thread (with each pass corresponding to one question), which cannot address the interactions well. In comparison, our model performs the extraction in one pass of the thread.

Inference algorithm. Inference is usually a time-consuming process for structured prediction. We design special inference algorithms, instead of general-purpose inference algorithms used in previous works (Cong et al., 2008; Ding et al., 2008), by taking advantage of special properties of our task. Specifically, we utilize two special properties of thread structure to reduce the inference (time) cost. First, *context* sentences and *question* sentences usually occur in the same post while *answer* sentences can only occur in the following posts. With this properties, we can greatly reduce *context* (or *answer*) candidate sets of a question, which results in a significant decrease in inference cost (Section 3). Second, *context* candidate set is usually much smaller than the number of sentences in a thread. This property enables our proposal to have an exact and efficient inference (Section 4.1). Moreover, an approximate inference algorithm is also given (Section 4.2).

Loss function. In practice, different application settings usually imply different requirements for system performance. For example, we expect a higher recall for the purpose of archiving questions but a higher precision for the purpose of retrieving questions. A flexible framework should be able to cope with various requirements. We employ structural Support Vector Machine (SVM) model that could naturally incorporate different loss functions (Section 5).

We use a real data set to evaluate our approach to extracting contexts and answers of questions. The experimental results show both the effectiveness and the flexibility of our approach.

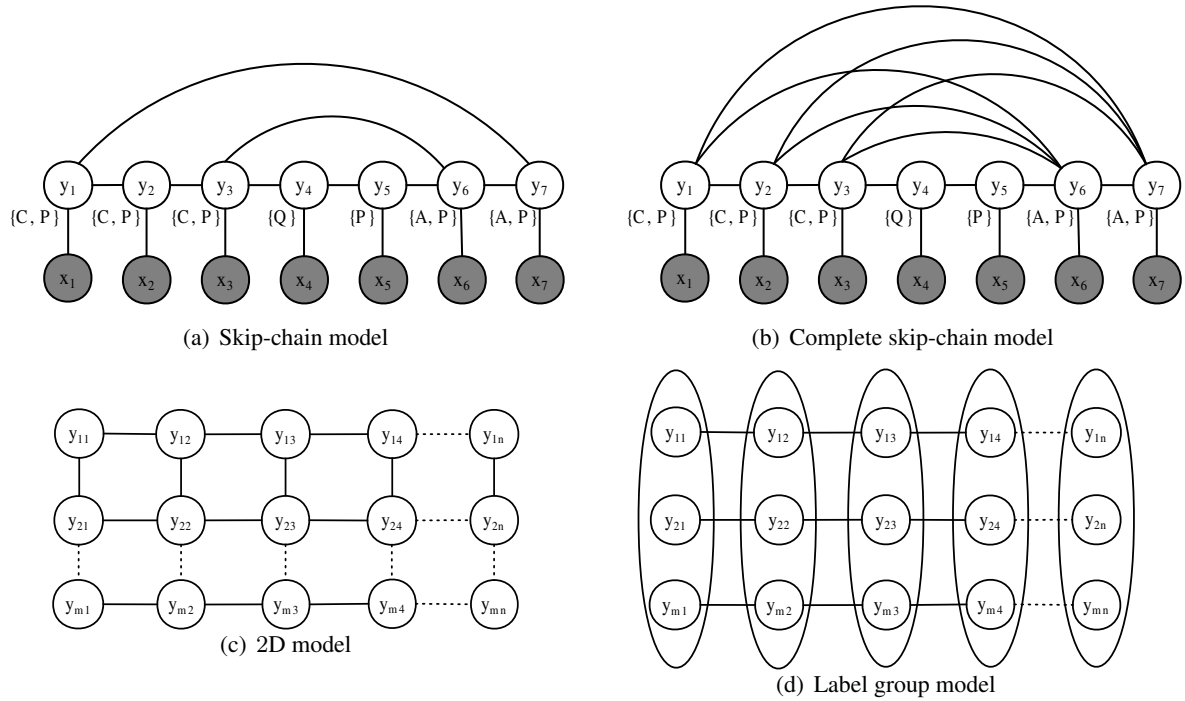
In the next section, we formalize the problem of context and answer extraction and introduce the structural model. In Sections 3, 4 and 5 we give the details of customizing structural model for our task. In Section 6, we evaluate our methods. In Section 7, we discuss the related work. Finally, we conclude this paper in Section 8.

2 Problem Statement

We first introduce our notations in Section 2.1 and then in Section 2.2 introduce how we model the problem of extracting contexts and answers for questions with a novel form of graphical representation. In Section 2.3 we introduce the structured model based on the new representation.

2.1 Notations

Assuming that a given thread contains p posts $\{p_1, \ldots, p_p\}$, which are authored by a set of users $\{u_1, \ldots, u_p\}$. The p posts can be further segmented into n sentences $\mathbf{x} = \{x_1, \ldots, x_n\}$. Among the n sentences, m *question* sentences $\mathbf{q} = \{x_{q_1}, \ldots, x_{q_m}\}$ have been identified. Our task is to identify the *context* sentences and the *answer* sentences for those m question sentences. More formally, we use four types of label $\{C, A, Q, P\}$ to stand for context, answer, question and plain labels. Then, our task is to predict an $m \times n$ label matrix $\mathbf{y} = (y_{ij})_{1 \le i \le m, 1 \le j \le n}$, except m elements $\{y_{1,q_1}, \ldots, y_{m,q_m}\}$ which correspond to (known) *question* labels. The element y_{ij} in label matrix $\mathbf{y}$ represents the role that the jth sentence plays for the ith question. We denote the ith row and jth column of the label matrix $\mathbf{y}$ by $\mathbf{y}_{i\cdot}$ and $\mathbf{y}_{\cdot j}$.



(a) Skip-chain model

(b) Complete skip-chain model

(c) 2D model

(d) Label group model

Figure 2: Structured models

2.2 Graphical Representation

Recently, Ding et al. (2008) use skip-chain and 2D Conditional Random Fields (CRFs) (Lafferty et al., 2001) to perform the relational learning for context and answer extraction. The skip-chain CRFs (Sutton and McCallum, 2004; Galley, 2006) model the long distance dependency between context and answer sentences and the 2D CRFs (Zhu et al., 2005) model the dependency between contiguous questions. The graphical representation of those two models are shown in Figures 2(a) and 2(c), respectively. Those two CRFs are both extensions of the linear chain CRFs for the sake of powerful relational learning. However, directly using the skip-chain and 2D CRFs without any customization has obvious disadvantages: (a) the skip-chain model does not model the *dependency between answer sentence and multiple context sentences*; and (b) the 2D model does not model the *dependency between non-contiguous questions*.

To better model the problem of extracting contexts and answers of questions, we propose two more comprehensive models, *complete skip-chain model* and *label group model* to improve the capability of the two previous models. These two models are shown in Figures 2(b) and 2(d).

In Figures 2(a) and 2(b), each label node is annotated with its allowed labels and the labels C, A, Q and P stand for context, answer, question and plain sentence labels, respectively. Note that the complete skip-chain model *completely* links each two context and answer candidates and the label group model combines the labels of one sentence into one *label group*.

2.3 Structured Model

Following the standard machine learning setup, we denote the input and output spaces by $\mathcal{X}$ and $\mathcal{Y}$, then formulate our task as learning a hypothesis function $h : \mathcal{X} \rightarrow \mathcal{Y}$ to predict a $\mathbf{y}$ when given $\mathbf{x}$. In this setup, $\mathbf{x}$ represents a thread of n sentences and m identified questions. $\mathbf{y}$ represents the $m \times n$ label matrix to be predicted.

Given a set of training examples, $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \in \mathcal{X} \times \mathcal{Y} : i = 1, \ldots, N\}$, we restrict ourselves to the supervised learning scenario. We focus on hypothesis functions that take the form $h(\mathbf{x}; \mathbf{w}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} \mathcal{F}(\mathbf{x}, \mathbf{y}; \mathbf{w})$ with discriminant function $\mathcal{F} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}$ where $\mathcal{F}(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y})$. As will be introduced in Section 4, we employ structural SVMs (Joachims et al., 2009) to find the optimal parameters $\mathbf{w}$. The structural SVMs have several competitive properties as CRFs. First, it follows from the maximum margin strategy, which has been shown with competitive or even better

performance (Tsochantaridis et al., 2005; Nguyen and Guo, 2007). Second, it allows flexible choices of loss functions to users. Moreover, in general, it has theoretically proved convergence in polynomial time (Joachims et al., 2009).

To use structural SVMs in relational learning, one needs to customize three steps according to specific tasks. The three steps are (a) *definition of joint feature mapping* for encoding relations, (b) *algorithm of finding the most violated constraint (inference)* for efficient trainings and (c) *definition of loss function* for flexible uses.

In the following Sections 3, 4 and 5, we describe the customizations of the three steps for our context and answer extraction task, respectively.

3 Encoding Relations

We use a joint feature mapping to model the relations between sentences in a thread. For context and answer extraction, the joint feature mapping can be defined as follows,

$$\Psi(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \Psi_n(\mathbf{x}, \mathbf{y}) \\ \Psi_h(\mathbf{x}, \mathbf{y}) \\ \Psi_v(\mathbf{x}, \mathbf{y}) \end{bmatrix},$$

where the sub-mappings $\Psi_n(\mathbf{x}, \mathbf{y})$, $\Psi_h(\mathbf{x}, \mathbf{y})$, and $\Psi_v(\mathbf{x}, \mathbf{y})$ encode three types of feature mappings, *node features*, *edge features* and *label group features*. The node features provide the basic information for the output labels. The edge features consist of the sequential edge features and skip-chain edge features for successive label dependencies. The label group features encode the relations within each label group.

Before giving the detail definitions of the sub-mappings, we first introduce *the context and answer candidate sets*, which will be used for the definitions and inferences. Each row of the label matrix $\mathbf{y}$ corresponds to one question. Assuming that the ith row $\mathbf{y}_{i\cdot}$ corresponds to the question with sentence index q_i, we thus have two candidate sets of contexts and answers for this question denoted by $\mathcal{C}$ and $\mathcal{A}$, respectively. We denote the post indices and the author indices for the n sentences as $\mathbf{p} = (p_1, \ldots, p_n)$ and $\mathbf{u} = (u_1, \ldots, u_n)$. Then, we can formally define the two candidate sets for the $\mathbf{y}_{i\cdot}$ as

$$\mathcal{C} = \left\{ c_j \;\middle|\; \underbrace{p_{c_j} = p_{q_i}}_{\text{In Question Post}}, \quad \underbrace{c_j \neq q_i}_{\text{Not Question Sentence}} \right\},$$

$$\mathcal{A} = \left\{ a_j \;\middle|\; \underbrace{p_{a_j} > p_{q_i}}_{\text{After Question Post}}, \quad \underbrace{u_{a_j} \neq u_{q_i}}_{\text{Not by the Same User}} \right\}.$$

In the following, we describe formally about the definitions of the three feature sub-mappings.

The node feature mapping $\Psi_n(\mathbf{x}, \mathbf{y})$ encodes the relations between sentence and label pairs, we define it as follows,

$$\Psi_n(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \psi_n(x_j, y_{ij}),$$

where $\psi_n(x_j, y_{ij})$ is a feature mapping for a given sentence and a label. It can be formally defined as follows,

$$\psi_n(x_j, y_{ij}) = \Lambda(y_{ij}) \otimes \phi_{q_i}(x_j), \qquad (1)$$

where $\otimes$ denotes a tensor product, $\phi_{q_i}(x_j)$ and $\Lambda(y_{ij})$ denote two vectors. $\phi_{q_i}(x_j)$ contains basic information for output label. $\Lambda(y_{ij})$ is a $0/1$ vector defined as

$$\Lambda(y_{ij}) = [\lambda_C(y_{ij}), \lambda_A(y_{ij}), \lambda_P(y_{ij})]^T,$$

where $\lambda_C(y_{ij})$ equal to one if $y_{ij} = C$, otherwise zero. The $\lambda_A(y_{ij})$ and $\lambda_P(y_{ij})$ are similarly defined. Thus, for example, writing out $\psi_n(x_j, y_{ij})$ for $y_{ij} = C$ one gets,

$$\psi_n(x_j, y_{ij}) = \begin{pmatrix} \phi_{q_i}(x_j) \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} \begin{matrix} \leftarrow \text{ context} \\ \leftarrow \text{ answer} \\ \leftarrow \text{ plain} \end{matrix}.$$

Note that the node feature mapping does not incorporate the relations between sentences.

The edge feature mapping $\Psi_h(\mathbf{x}, \mathbf{y})$ is used to incorporate two types of relation, the relation between successive sentences and the relation between context and answer sentences. It can be defined as follows,

$$\Psi_h(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \Psi_{hn}(\mathbf{x}, \mathbf{y}) \\ \Psi_{hc}(\mathbf{x}, \mathbf{y}) \end{bmatrix},$$

where $\Psi_{hn}(\mathbf{x}, \mathbf{y})$ and $\Psi_{hc}(\mathbf{x}, \mathbf{y})$ denote the two types of feature mappings corresponding to sequential edges and skip-chain edges, respectively. Their formal definitions are given as follows,

$$\Psi_{hn}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} \sum_{j=1}^{n-1} \psi_{hn}(x_j, x_{j+1}, y_{ij}, y_{i,j+1}),$$

Descriptions	Dimensions
$\psi_{q_i}(x_j)$ (32 dimensions) in $\Psi_n(\mathbf{x}, \mathbf{y})$	
The cosine, WordNet and KL-divergence similarities with the question x_{q_i}	3
The cosine, WordNet and KL-divergence similarities with the questions other than x_{q_i}	3
The cosine, WordNet and KL-divergence similarities with previous and next sentences	6
Is this sentence x_j exactly x_{q_i} or one of the questions in $\{x_{q_1}, \ldots, x_{q_m}\}$?	2
Is this sentence x_j in the three beginning sentences?	3
The relative position of this sentence x_j to questions	4
Is this sentence x_j share the same author with the question sentence x_{q_i}?	1
Is this sentence x_j in the same post with question sentences?	2
Is this sentence x_j in the same paragraph with question sentences?	2
The presence of greeting (e.g., "hi") and acknowledgement words in this sentence x_j	2
The length of this sentence x_j	1
The number of nouns, verbs and pronouns in this sentence x_j, respectively	3
$\Psi_h(\mathbf{x}, \mathbf{y})$ (704 dimensions)	
For $\Psi_{hn}(\mathbf{x}, \mathbf{y})$, the above 32 dimension features w.r.t. $4 \times 4 = 16$ transition patterns	512
For $\Psi_{hc}(\mathbf{x}, \mathbf{y})$, 12 types of pairwise or merged similarities w.r.t. 16 transition patterns	192
$\Psi_v(\mathbf{x}, \mathbf{y})$ (32 dimensions)	
The transition patterns for any two non-contiguous labels in a label group	16
The transition patterns for any two contiguous labels in a label group	16

Table 1: Feature descriptions and demisions

$$\Psi_{hc}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} \underbrace{\sum_{j \in \mathcal{C}} \sum_{k \in \mathcal{A}}}_{\text{Complete Edges}} \psi_{hc}(x_j, x_k, y_{ij}, y_{ik}),$$

$$\begin{aligned} &\psi_{hn}(x_j, x_{j+1}, y_{ij}, y_{i,j+1}) \\ &= \Lambda(y_{ij}, y_{i,j+1}) \otimes \phi_{hn}(x_j, x_{j+1}, y_{ij}, y_{i,j+1}), \end{aligned}$$

$$\begin{aligned} &\psi_{hc}(x_j, x_k, y_{ij}, y_{ik}) \\ &= \Lambda(y_{ij}, y_{ik}) \otimes \psi_{hc}(x_j, x_k, y_{ij}, y_{ik}) \end{aligned}$$

where $\Lambda(y_{ij}, y_{ik})$ is a 16-dimensional vector. It indicates all 4×4 pairwise transition patterns of four types of labels, the context, answer, question and plain. Note that apart from previous work (Ding et al., 2008) we use *complete skip-chain (context-answer) edges* in $\Psi_{hc}(\mathbf{x}, \mathbf{y})$.

The label group feature mapping $\Psi_v(\mathbf{x}, \mathbf{y})$ is defined as follows,

$$\Psi_v(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{n} \psi_v(x_j, \mathbf{y}_{\cdot j}),$$

where $\psi_v(x_j, \mathbf{y}_{\cdot j})$ encodes each label group pattern into a vector.

The detail descriptions and vector dimensions of the used features are listed in Table 1.

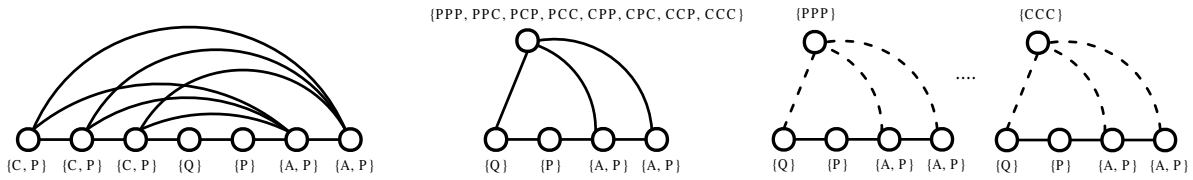
4 Structural SVMs and Inference

Given a training set $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \in \mathcal{X} \times \mathcal{Y} : i = 1, \ldots, N\}$, we use the structural SVMs (Taskar et al., 2003; Tsochantaridis et al., 2005; Joachims et al., 2009) formulation, as shown in Optimization Problem 1 (OP1), to learn a weight vector $\mathbf{w}$.

OP 1 *(1-Slack Structural SVM)*

$$\begin{aligned} \min_{\mathbf{w}, \xi \geq 0} \quad & \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N}\xi \\ \textit{s.t.} \quad & \forall(\bar{\mathbf{y}}^{(1)}, \ldots, \bar{\mathbf{y}}^{(N)}) \in \mathcal{Y}^n, \\ & \frac{1}{N}\mathbf{w}^T \sum_{i=1}^{N} [\Psi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \Psi(\mathbf{x}^{(i)}, \bar{\mathbf{y}}^{(i)})] \\ & \geq \frac{1}{N} \sum_{i=1}^{N} \Delta(\mathbf{y}^{(i)}, \bar{\mathbf{y}}^{(i)}) - \xi, \end{aligned}$$

where ξ is a slack variable, $\Psi(\mathbf{x}, \mathbf{y})$ is the joint feature mapping and $\Delta(\mathbf{y}, \bar{\mathbf{y}})$ is the loss function that measures the loss caused by the difference between $\mathbf{y}$ and $\bar{\mathbf{y}}$. Though OP1 is already a quadratic optimization problem, directly using off-the-shelf quadratic optimization solver will fail, due to the large number of constraints. Instead, a cutting plane algorithm is used to efficiently solve this problem. For the details of the



(a) Original graph (b) Transformed graph (c) Decomposed graph

Figure 3: The equivalent transform of graphs

Algorithm 1 Exact Inference Algorithm

1: Input: $(\mathcal{C}_i, \mathcal{A}_i)$ for each q_i, $\mathbf{w}$, $\mathbf{x}$, $\mathbf{y}$
2: **for** $i \in \{1, \ldots, m\}$ **do**
3: **for** $\mathcal{C}_s \subseteq \mathcal{C}_i$ **do**
4: $[R(\mathcal{C}_s), \bar{\mathbf{y}}_{i.}(\mathcal{C}_s)] \leftarrow \text{Viterbi}(\mathbf{w}, \mathbf{x}; \mathcal{C}_s)$
5: **end for**
6: $\mathcal{C}_s^* = \arg\max_{\mathcal{C}_s \subseteq \mathcal{C}_i} R(\mathcal{C}_s)$
7: $\bar{\mathbf{y}}_{i.}^* = \bar{\mathbf{y}}_{i.}(\mathcal{C}_s^*)$
8: **end for**
9: **return** $\bar{\mathbf{y}}^*$

structural SVMs, please refer to (Tsochantaridis et al., 2005; Joachims et al., 2009).

The most essential and time-consuming step in structural SVMs is *finding the most violated constraint*, which is equivalent to solve

$$\arg\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{y}) + \Delta(\mathbf{y}^{(i)}, \mathbf{y}). \quad (2)$$

Without the ability to efficiently find the most violated constraint, the cutting plane algorithm is not tractable.

In the next sub-sections, we introduce the algorithms for finding the most violated constraint, also called loss-augmented inference. The algorithms are essential for the success of customizing structural SVMs to our problem.

4.1 Exact Inference

The exact inference algorithm is designed for a simplified model with two sub-mappings Ψ_n and Ψ_h, except Ψ_v.

One naive approach to finding the most violated constraint for the simplified model is to enumerate all the $2^{|\mathcal{C}|+|\mathcal{A}|}$ cases for each row of the label matrix. However, it would be intractable for large candidate sets.

An important property is that the context candidate set is usually much smaller than the whole number of sentences in a thread. This property enables us to design efficient and exact inference algorithm by transforming from the original graph representation in Figure 2 to the graphs in Figure 3. This graph transform merges all the nodes in the context candidate set $\mathcal{C}$ to one node with $2^{|\mathcal{C}|}$ possible labels.

We design an exact inference algorithm in Algorithm 1 based on the graph in Figure 3(c). The algorithm can be summarized in three steps: (1) enumerate all the $2^{|\mathcal{C}|}$ possible labels[1] for the merged node (line 3). (2) For each given label of the merged node, perform the Viterbi algorithm (Rabiner, 1989) on the decomposed graph (line 4) and store the Viterbi algorithm outputs in R and $\hat{\mathbf{y}}_{i.}$. (3) From the $2^{|\mathcal{C}|}$ Viterbi algorithm outputs, select the one with highest score as the output (lines 6 and 7).

The use of the Viterbi algorithm is assured by the fact that there exists certain equivalence between the decomposed graph (Figure 3(c)) and a linear chain. By fixing the the label of the merged node, we could remove the dashed edges in the decomposed graph and regard the rest graph as a linear chain, which results in the Viterbi decoding.

4.2 Approximate Inference

The exact inference cannot handle the complete model with three sub-mappings, Ψ_n, Ψ_h, and Ψ_v, since the label group defeats the graph transform in Figure 3. Thus, we design two approximate algorithms by employing *undergenerating* and *overgenerating* approaches (Finley and Joachims, 2008).

First, we develop an *undergenerating* local greedy search algorithm shown in Algorithm 2. In the algorithm, there are two loops, inner and outer loops. The outer loop terminates when no labels change (steps 3-11). The inner loop enumerates the whole label matrix and greedily determines each label (step 7) by maximizing the Equation (2). Since the whole algorithm terminates only if

[1] Since the merged node is from context candidate set $\mathcal{C}$, enumerating its label is equivalent to enumerating subsets $\mathcal{C}_s$ of the candidate set $\mathcal{C}$

Algorithm 2 Greedy Inference Algorithm

Input: $\mathbf{w}$, $\mathbf{x}$, $\mathbf{y}$
initialize solution: $\bar{\mathbf{y}} \leftarrow \mathbf{y}_0$
repeat
  $\mathbf{y}' \leftarrow \bar{\mathbf{y}}$
  for $i \in \{1, \ldots, m\}$ **do**
    for $j \in \{1, \ldots, n\}$ **do**
      $\bar{y}^*_{ij} \leftarrow \arg\max_{\bar{y}_{ij}} \mathbf{w}^T \Psi(\mathbf{x}, \bar{\mathbf{y}}) + \triangle(\mathbf{y}, \bar{\mathbf{y}})$
      $\bar{y}_{ij} \leftarrow \bar{y}^*_{ij}$
    end for
  end for
until $\bar{\mathbf{y}} = \mathbf{y}'$
$\bar{\mathbf{y}}^* \leftarrow \bar{\mathbf{y}}$
return $\bar{\mathbf{y}}^*$

the label matrix does not change during the last outer loop. This indicates that at least a local optimal solution is obtained.

Second, an *overgenerating* method can be designed by using linear programming relaxation (Finley and Joachims, 2008). To save the space, we skip the details of this algorithm here.

5 Loss Functions

Structural SVMs allow users to customize the loss function $\triangle : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathcal{R}$ according to different system requirements. In this section, we introduce the loss functions used in our work.

Basic loss function. The simplest way to quantify the prediction quality is counting the number of wrongly predicted labels. Formally,

$$\triangle_b(\mathbf{y}, \bar{\mathbf{y}}) = \sum_{i=1}^{m} \sum_{j=1}^{n} I[y_{ij} \neq \bar{y}_{ij}], \qquad (3)$$

where $I[.]$ is an indicative function that equals to one if the condition holds and zero otherwise.

Recall-vs-precision loss function. In practice, we may place different emphasis on recall and precision according to application settings. We could include this preference into the model by defining the following loss function,

$$\begin{aligned} \triangle_p(\mathbf{y}, \bar{\mathbf{y}}) \quad = \quad & \sum_{i=1}^{m} \sum_{j=1}^{n} I[y_{ij} \neq P, \bar{y}_{ij} = P] \cdot c_r \\ & + I[y_{ij} = P, \bar{y}_{ij} \neq P] \cdot c_p. \end{aligned} \qquad (4)$$

This function penalizes the wrong prediction decreasing recall and that decreasing precision with two weights c_r and c_p respectively. Specifically, we denote the loss function with $c_p/c_r = 2$ and that with $c_r/c_p = 2$ by $\triangle^p_p$ and $\triangle^r_p$, respectively.

Various types of loss function can be defined in a similar fashion. To save the space, we skip the definitions of other loss functions and only use the above two types of loss functions to show the flexibility of our approach.

Items in the data set	#items
Thread	515
Post	2,035
Sentence	8,500
question annotation	1,407
context annotation	1,962
answer annotation	4,652
plain annotation	18,198

Table 2: The data statistics

6 Experiments

6.1 Experimental Setup

Corpus. We made use of the same data set as introduced in (Cong et al., 2008; Ding et al., 2008). Specifically, the data set includes about 591 threads from the forum TripAdvisor[2]. Each sentence in the threads is tagged with the labels 'question', 'context', 'answer', or 'plain' by two annotators. We removed 76 threads that have no question sentences or more than 40 sentences and 6 questions. The remaining 515 forum threads form our data set.

Table 2 gives the statistics on the data set. On average, each thread contains 3.95 posts and 2.73 questions, and each question has 1.39 context sentences and 3.31 answer sentences. Note that the number of annotations is much larger than the number of sentences because one sentence can be annotated with multiple labels.

Experimental Details. In all the experiments, we made use of linear models for the sake of computational efficiency. As a preprocessing step, we normalized the value of each feature value into the interval $[0, 1]$ and then followed the heuristic used in SVM-light (Joachims, 1998) to set C to $1/||x||^2$, where $||x||$ is the average length of input samples (in our case, sentences). The tolerance parameter ϵ was set to 0.1 (the value also used in (Cai

[2]TripAdvisor (`http://www.tripadvisor.com/ForumHome`) is one of the most popular travel forums

and Hofmann, 2004)) in all the runs of the experiments.

Evaluation. We calculated the standard precision (P), recall (R) and F_1-score (F_1) for both tasks (context extraction and answer extraction). All the experimental results were obtained through 5-fold cross validation.

6.2 Baseline Methods

We employed binary SVMs (B-SVM), multiclass SVMs (M-SVM), and C4.5 (Quinlan, 1993) as our baseline methods:

B-SVM. We trained two binary SVMs for context extraction (context vs. non-context) and answer extraction (answer vs. non-answer), respectively. We used the feature mapping $\phi_{q_i}(x_j)$ defined in Equation (1) while training the binary SVM models.

M-SVM. We extended the binary SVMs by training multiclass SVMs for three category labels (*context*, *answer*, *plain*).

C4.5. This decision tree algorithm solved the same classification problem as binary SVMs and made use of the same set of features.

Method	$\triangle_b$	P (%)	R (%)	F_1 (%)
Context Extraction				
C4.5	–	74.2	68.7	71.2
B-SVM	–	78.3	72.2	74.9
M-SVM	–	68.0	77.6	72.1
S-SVM	8.86	75.6	71.7	73.4
S-SVM-H	8.60	77.5	75.5	76.3
S-SVM-HC*	8.65	77.9	74.1	75.8
S-SVM-HC	8.62	77.5	75.2	76.2
S-SVM-HCV*	8.08	79.5	79.6	79.5
S-SVM-HCV	**7.98**	79.7	80.2	**79.9**
Answer Extraction				
C4.5	–	61.3	45.2	51.8
B-SVM	–	69.7	42.0	51.8
M-SVM	–	63.2	51.5	55.8
S-SVM	8.86	67.0	48.0	55.6
S-SVM-H	8.60	66.9	49.7	56.7
S-SVM-HC*	8.65	66.5	49.4	56.4
S-SVM-HC	8.62	65.7	51.5	57.4
S-SVM-HCV*	8.08	65.5	58.7	61.7
S-SVM-HCV	**7.98**	65.1	61.2	**63.0**

Table 3: The effectiveness of our approach

6.3 Modeling Sentence Relations and Question Interactions

We demonstrate in Table 3 that our approach can make use of the three types of relation among sentences well to boost the performance.

In Table 3, S-SVM represents the structural SVMs only using the node features $\Psi_n(\mathbf{x}, \mathbf{y})$. The suffixes *H*, *C*, and *V* denote the models using *horizontal sequential edges*, *complete skip-chain edges* and *vertical label groups*, respectively. The suffixes *C** and *V** denote the models using *incomplete skip-chain edges* and *vertical sequential edges* proposed in (Ding et al., 2008), as shown in Figures 2(a) and 2(c). All the structural SVMs were trained using basic loss function Δ_b in Equation (3). From Table 3, we can observe the following advantages of our approaches.

Overall improvement. Our structural approach steadily improves the extraction as more types of relation (corresponding to more types of edge) are included. The best results obtained by using the three types of relation together improve the baseline methods binary SVMs by about 6% and 20% in terms of F_1 values for context extraction and answer extraction, respectively.

The usefulness of relations. The relations encoded by horizontal sequential edges and label groups are useful for both context extraction and answer extraction. The relation encoded by complete skip-chain edges is useful for answer extraction. The complete skip-chain edges not only avoid preprocessing but also boost the performance when compared with the preprocessed skip-chain edges. The label groups improve the vertical sequential edges.

Interactions among questions. The interactions encoded by label groups are especially useful. We conducted significance tests (sign test) on the experimental results. The test result shows that S-SVM-HCV outperforms all the other methods without vertical edges statistically significantly (p-value < 0.01). Our proposed graphical representation in Figure 2(d) eases us to model the complex interactions. In comparison, the 2D model in Figure 2(c) used in previous work (Ding et al., 2008) can only model the interaction between adjacent questions.

6.4 Loss Function Results

We report in Table 4 the comparison between structural SVMs using different loss functions. Note that Δ_p^p prefers precision and Δ_p^r prefers recall. From Table 4, we can observe that the experimental results also exhibit this kind of system

Method	P (%)	R (%)	F_1 (%)
Context Extraction			
S-SVM-HCV-$\triangle_b$	79.7	80.2	79.9
S-SVM-HCV-$\triangle_p^p$	**82.0**	70.3	75.6
S-SVM-HCV-$\triangle_p^r$	75.7	**84.2**	79.7
Answer Extraction			
S-SVM-HCV-$\triangle_b$	65.1	61.2	63.0
S-SVM-HCV-$\triangle_p^p$	**71.8**	52.2	60.2
S-SVM-HCV-$\triangle_p^r$	61.8	**66.1**	63.7

Table 4: The use of different loss functions

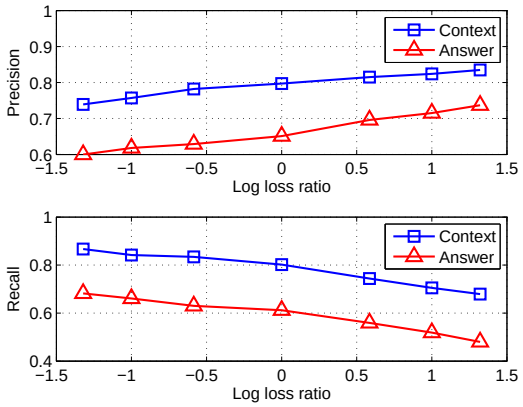


Figure 4: Balancing between precision and recall

preference. Moreover, we further demonstrate the capability of the loss function Δ_p in Figure 4. The curves are achieved by varying the ratio between two parameters c_p/c_r in Equation (4). The curves confirm our intuition: when $\log(c_p/c_r)$ becomes larger, the precisions increase but the recalls decrease and vice versa.

7 Related work

Previous work on extracting questions, answers and contexts is most related with our work. Cong et al. (2008) proposed a supervised approach for question detection and an unsupervised approach for answer detection without considering contexts. Ding et al. (2008) used CRFs to detect contexts and answers of questions from forum threads.

Some researches on summarizing discussion threads and emails are related to our work, too. Zhou and Hovy (2005) segmented internet relay chat, clustered segments into sub-topics, and identified responding segments of the first segment in each sub-topic by assuming the first segment to be focus. In (Nenkova and Bagga, 2003; Wan and McKeown, 2004; Rambow et al., 2004), email summaries were organized by extracting overview sentences as discussion issues. The work (Shrestha and McKeown, 2004) used RIPPER as a classifier to detect interrogative questions and their answers then used the resulting question and answer pairs as summaries. We also note the existing work on extracting knowledge from discussion threads. Huang et al. (2007) used SVMs to extract input-reply pairs from forums for chatbot knowledge. Feng et al. (2006) implemented a discussion-bot which used cosine similarity to match students' query with reply posts from an annotated corpus of archived threaded discussions.

Moreover, extensive researches have been done within the area of question answering (Burger et al., 2006; Jeon et al., 2005; Harabagiu and Hickl, 2006; Cui et al., 2005; Dang et al., 2006). They mainly focused on using sophisticated linguistic analysis to construct answer from a large document collection.

8 Conclusion and Future Work

We have proposed a new form of graphical representation for modeling the problem of extracting contexts and answers of questions from online forums and then customized structural SVM approach to solve it.

The proposed graphical representation is able to naturally express three types of relation among sentences: relation between successive sentences, relation between context sentences and answer sentences, and relation between multiple labels for one sentence. The representation also enables us to address interactions among questions. We also developed the inference algorithms for the structural SVM model by exploiting the special structure of thread discussions.

Experimental results on a real data set show that our approach significantly improves the baseline methods by effectively utilizing various types of relation among sentences.

Our future work includes: (a) to summarize threads and represent the forum threads in question-context-answer triple, which will change the organization of online forums; and (b) to enhance QA services (e.g., Yahoo! Answers) by the contents extracted from online forums.

Acknowledgement

The authors would like to thank the anonymous reviewers for their comments to improve this paper.

References

John Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda Harabagiu, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, George Miller, Dan Moldovan, Bill Ogden, John Prager, Ellen Riloff, Amit Singhal, Rohini Shrihari, Tomek Strzalkowski, Ellen Voorhees, and Ralph Weishedel. 2006. Issues, tasks and program structures to roadmap research in question and answering (qna). *ARAD: Advanced Research and Development Activity (US)*.

Lijuan Cai and Thomas Hofmann. 2004. Hierarchical document categorization with support vector machines. In *Proceedings of CIKM*, pages 78–87.

Gao Cong, Long Wang, Chin-Yew Lin, and Young-In Song. 2008. Finding question-answer pairs from online forums. In *Proceedings of SIGIR*, pages 467–474.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of SIGIR*, pages 400–407.

Hoa Dang, Jimmy Lin, and Diane Kelly. 2006. Overview of the trec 2006 question answering track. In *Proceedings of TREC*, pages 99–116.

Shilin Ding, Gao Cong, Chin-Yew Lin, and Xiaoyan Zhu. 2008. Using conditional random field to extract contexts and answers of questions from online forums. In *Proceedings of ACL*, pages 710–718.

Donghui Feng, Erin Shaw, Jihie Kim, and Eduard H. Hovy. 2006. An intelligent discussion-bot for answering student queries in threaded discussions. In *Proceedings of IUI*, pages 171–177.

Thomas Finley and Thorsten Joachims. 2008. Training structural SVMs when exact inference is intractable. In *Proceedings of ICML*, pages 304–311.

Michel Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 364–372.

Sanda M. Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of ACL*, pages 905–912.

Jizhou Huang, Ming Zhou, and Dan Yang. 2007. Extracting chatbot knowledge from online discussion forums. In *Proceedings of IJCAI*, pages 423–428.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of CIKM*, pages 84–90.

Thorsten Joachims, Thomas Finley, and Chun-Nam Yu. 2009. Cutting-plane training of structural SVMs. *Machine Learning*.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of ECML*, pages 137–142.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Ani Nenkova and Amit Bagga. 2003. Facilitating email thread access by extractive summary generation. In *Proceedings of RANLP*, pages 287–296.

Nam Nguyen and Yunsong Guo. 2007. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of ICML*, pages 681–688.

John Quinlan. 1993. *C4.5: programs for machine learning*. Morgan Kaufmann Publisher Incorporation.

Lawrence Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of IEEE*, pages 257–286.

Owen Rambow, Lokesh Shrestha, John Chen, and Chirsty Lauridsen. 2004. Summarizing email threads. In *Proceedings of HLT-NAACL*, pages 105–108.

Lokesh Shrestha and Kathleen McKeown. 2004. Detection of question-answer pairs in email conversations. In *Proceedings of COLING*, pages 889–895.

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. Technical Report 04-49.

Benjamin Taskar, Carlos Guestrin, and Daphne Koller. 2003. Max-margin markov networks. In *Advances in Neural Information Processing Systems 16*. MIT Press.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

Stephen Wan and Kathy McKeown. 2004. Generating overview summaries of ongoing email thread discussions. In *Proceedings of COLING*, pages 549–555.

Liang Zhou and Eduard Hovy. 2005. Digesting virtual "geek" culture: The summarization of technical internet relay chats. In *Proceedings of ACL*, pages 298–305.

Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2005. 2d conditional random fields for web information extraction. In *Proceedings of ICML*, pages 1044–1051.

Mining Search Engine Clickthrough Log for Matching *N*-gram Features

Huihsin Tseng, Longbin Chen, Fan Li, Ziming Zhuang,
Lei Duan, Belle Tseng
Yahoo! Inc., Santa Clara, CA 95054
{huihui,longbin,fanli,ziming,leiduan,belle}@yahoo-inc.com

Abstract

User clicks on a URL in response to a query are extremely useful predictors of the URL's relevance to that query. Exact match click features tend to suffer from severe data sparsity issues in web ranking. Such sparsity is particularly pronounced for new URLs or long queries where each distinct query-url pair will rarely occur. To remedy this, we present a set of straightforward yet informative query-url *n*-gram features that allows for generalization of limited user click data to large amounts of unseen query-url pairs. The method is motivated by techniques leveraged in the NLP community for dealing with unseen words. We find that there are interesting regularities across queries and their preferred destination URLs; for example, queries containing "form" tend to lead to clicks on URLs containing "pdf". We evaluate our set of new query-url features on a web search ranking task and obtain improvements that are statistically significant at a *p*-value < 0.0001 level over a strong baseline with exact match clickthrough features.

1 Introduction

Clickthrough logs record user click behaviors, which are a critical source for improving search relevance (Bilenko and White, 2008; Radlinski et al., 2007; Agichtein and Zheng, 2006; Lu et al. 2006). Previous work (Agichtein et al., 2006) demonstrated that clickthrough features (e.g., IsNextClicked and IsPreviousClicked) can lead to substantial improvements in relevance. Such features summarize query-specific user interactions on a search engine. One commonly used clickthrough feature is generated based on the following observation: if a URL receives a large number of first and last clicks across many user sessions, then it indicates that this URL might be a strongly preferred destination of a query. For example, when a user searches for "yahoo", they tend to only click on the URL www.yahoo.com rather than other alternatives. This results in www.yahoo.com being the first and last clicked URL for the query. We refer to such behavior as being navigational clicks (**NavClicks**). Features that use exact query and URL string matches (e.g., NavClick, IsNextClicked and IsPreviousClicked) are referred to as exact match features (**ExactM**) for the remainder of this paper.

The coverage of ExactM features is sparse, especially for long queries and new URLs. Many long queries are either unique or very low frequency. Hence, the improvements from ExactM features are limited to the more popular queries. In addition, ExactM features tend to be weighted heavily in the ranking of results when they are available. This introduces a bias where the ranking models tend to strongly favor older URLs over new URLs even when the latter otherwise appear to be more relevant.

By inspecting the clickthrough logs, we observed that unseen query-url pairs are often composed of informative previously observed subsequences. Specifically, we saw that query *n*-grams can be correlated with sequences of URL *n*-grams. For example, we find that there are interesting regularities across queries and URLs, such as queries containing "form" tending to lead to clicks on URLs containing "pdf". This strongly motivates the adoption of an approach similar to the Natural Language Processing (**NLP**) technique of using *n*-grams to deal with unseen words. For example, part-of-speech tagging (Brants, 2000) and parsing (Klein and Manning, 2003) both require dealing with unknown words. By using *n*-gram substrings, novel items can be dealt with using any informative substrings they contain that were actually observed in the training data.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 524–533,
Singapore, 6-7 August 2009.

The remainder of the paper is organized as follows. In Section 2, we introduce our overall methodology. Section 2.1 presents a data mining method for building a query-url *n*-gram dictionary, Section 2.2 describes the new ranking features in detail. In section 3, we present our experimental results. Section 4 discusses related work, and Section 5 summarizes the contribution of this work.

2 Methodology

This section describes the detailed methodology used in generating the query-url *n*-gram features. Our features require association scores to be previously calculated, and, hence, we first introduce a data mining approach that is used to build an association dictionary in Section 2.1. Then, we present the procedure used to generate the query-url *n*-gram features that use the dictionary in Section 2.2.

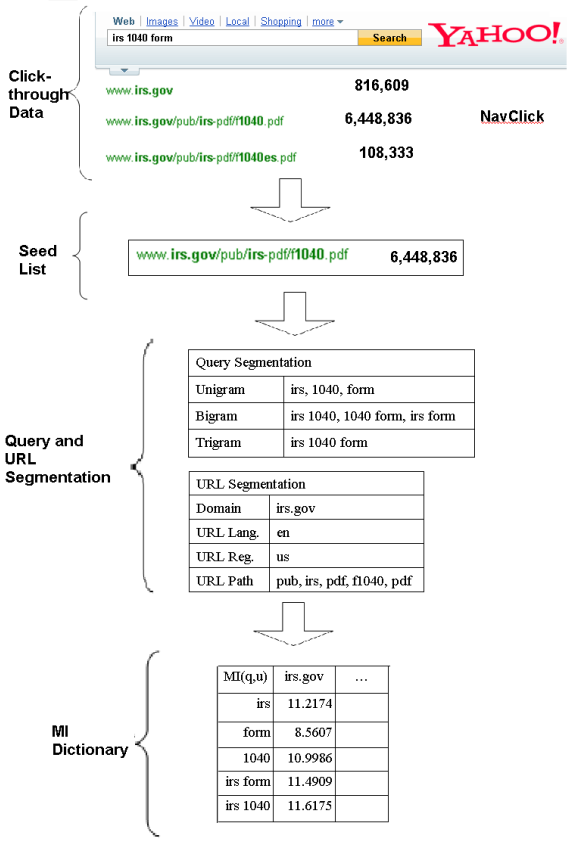


Figure 1: Steps to build a query-url *n*-gram dictionary

2.1 Data Mining on a Query-URL *N*-gram Dictionary

The steps involved in building the dictionary are shown in Figure 1. We first collect seed query-url pairs from clickthrough data based on NavClicks. The queries and URLs from the collected pairs are tokenized and converted into a collection of paired query-url *n*-grams. For each pair, we calculate the mutual information of the query *n*-gram and its corresponding URL *n*-gram. For our experiment, we collect a total of more than 15M seed pairs and 0.5B query-url *n*-gram pairs using six months of query log data. The details are described in the following sections.

2.1.1 Seed List

We identify the seed list based on characteristic user click behavior. Given a query, we select the URL with the most NavClicks as compared to other URLs returned. During data collection, the rank positions of the top 5 URLs were shuffled to avoid the position bias. We aggregate NavClicks for a URL occurring in these positions in order to both obtain more click data and to avoid the position bias issue discussed in Dupret and Piwowarski (2008) and Craswell et al. (2008).

For example, in Figure 1, the numbers of NavClicks for the top three URLs are shown. The URL www.irs.gov/pub/irs-pdf/f1040.pdf receives the largest number of NavClicks, and, therefore, it is used to create the query-url pair:

[irs 1040 form, www.irs.gov/pub/irs-pdf/f1040.pdf]

2.1.2 Query and URL Segmentation

We segment the seed pairs to *n*-gram pairs in order to increase the coverage beyond that of ExactM click features. Within NLP, *n*-grams are typically extracted such that words that are adjacent in the original sequence are also adjacent in the extracted *n*-grams. Furthermore, we attempt to achieve additional generalization by using skip *n*-grams (Lin and Och, 2004). This means we not only extract *n*-grams for adjacent terms but also for sequences that leave out intermediate terms. This is motivated by the observation that the semantics of user queries is often preserved even when some intermediate terms are removed. The details of the segmentation methods are described below.

2.1.2.1 Query Segmentation

Prior to query segmentation, we normalize raw queries by replacing punctuations with spaces. Queries are then segmented into a sequence of

space delimited tokens. From these, we extract all possible query *n*-grams and skip *n*-grams for *n* smaller than or equal to three (i.e., all unigrams, bigrams, and trigrams). For example, given the sequence "irs 1040 form" the adjacent bigrams would be "irs 1040" and "1040 form". With skip *n*-grams we also extract "irs form" as shown in Table 1. We do not use *n*-grams longer than 3 in order to avoid problems with overfitting. We will refer to this segmentation method as **Affix Segmentation**.

Table 1: An Example of Affix Segmentation

N-gram	Affix Segmentation
Unigram	irs, 1040, form
Bigram	irs 1040, 1040 form, irs form
Trigram	irs 1040 form

2.1.2.2 URL Segmentation

As shown in Table 2, after the queries are segmented, URLs are categorized into four groups: domain, URL language, URL region and URL path. In general, a URL is delimited by punctuation characters such as "?", "." "/", and "=".

Table 2: An Example of URL Segmentation

URL Groups	Example
Domain	irs.gov
URL language	en
URL region	us
URL path	pub, irs, pdf, f1040, pdf

The domain group includes one domain token, for example, irs.gov. Although domains could be divided into multiple *n*-grams, we treat them as a single unit, with the exception of encoded language and region information.

The language and region groups are based on the language or region part of the URL *n*-grams such as the suffixes ".en" and ".de". The language and region of a URL *n*-gram are identified by a table look-up method. The table is created based on the information available at en.wikipedia.org/wiki/List_of_ISO_639-1_codes and en.wikipedia.org/wiki/ISO_3166. When there is no clear language or region URL *n*-gram, we use English (en) as the default language and United States (us) as the default region.

2.1.3 Calculation of Mutual Information

After query and URL *n*-grams are extracted, we calculate mutual information (Gale and Church, 1991) to determine the degree of association between the *n*-grams. The definition of query-url *n*-gram mutual information (MI) is given in Equation 1.

$$\mathrm{MI}(q,u) = \log_2 \frac{\mathrm{Freq}(q,u)}{\mathrm{Freq}(q)\,\mathrm{Freq}(u)} \quad (1)$$

Here q corresponds to a query *n*-gram and u corresponds to a URL *n*-gram. Freq (q) is the count of q in the seed list normalized by the total number of q. Freq (u) is the count of u normalized by the total number of u. Freq (q, u) is the count of q and u that co-occurred in a full query-url pair normalized by the total number of q and u. A pair will be assigned to a MI score of zero if the items occur together no more than expected by chance, under the assumption that the two items are statistically independent. When a pair occurs more than is expected by chance, the MI score is positive. On the other hand, if a pair occurs together less than is expected by chance, the mutual information score is negative. In order to increase the confidence of the MI scores, we remove all *n*-grams with less than 3 occurrences in the seed list, and assign a zero MI score for any pairs involving these *n*-grams. No smoothing is applied.

This scoring scheme fits well with the association properties we would like to have for our query-url *n*-gram click features. If a query *n*-gram cues for a certain URL through one of its *n*-grams, the feature will take on a positive value. Similarly, if a query *n*-gram cues against a certain URL, the feature will take on a negative value.

2.1.4 Analysis of Query-URL *N*-gram Association

By examining our dictionary, we observed a number of pairs that are interesting from a relevance ranking perspective. To illustrate, we present four examples of *n*-gram pairs and intuitively explore the nature of the *n*-gram associations in the dictionary.

Table 3: Examples of MI Scores

Query *n*-gram	URL *n*-gram	MI score
"iphone"	apple.com	8.7713
"iphone"	amazon.com	-0.1555
"iphone plan"	att.com	11.5388
"iphone plan"	apple.com	8.9676

First, let's examine the association between query *n*-grams and URL *n*-grams for the queries

"iphone" and "iphone plan". Notice that the query unigram "iphone" is strongly associated with apple.com, but negatively associated with amazon.com. This can be explained by the fact that "iphone" as a product is not only developed by Apple but also strongly associated with the Apple brand. In contrast, while Amazon.com sells iphones, it also sells a large variety of other products, thus is not regarded as a very authoritative source of information about the "iphone". However, by adding additional context, the most preferred URL according to MI can change. The two examples in the bottom of Table 3 illustrate the URL preferences for the query bigram "iphone plan". While apple.com is still a strongly preferred destination, there is a much stronger preference for att.com. This preference follows since apple.com has more product information on the "iphone" while the information provided by att.com will be more targeted at visitors who want to explore what rate plans are available.

Second, Table 4 shows the association between "kimo", ".tw" and ".us". "Kimo" was a Taiwanese start-up acquired by Yahoo!. The mutual information scores accurately reflect the association between the query *n*-gram and region ids.

Table 4: Example of MI Scores

Query *n*-gram	URL *n*-gram	MI score
"kimo"	tw (taiwan)	12.8303
"kimo"	us (united states)	0.7209

Third, Table 5 shows the association between "kanji", and URLs with Language identification of "Japanese", "Chinese" and "English". "Kanji" means "Chinese" in Japanese. Since queries containing "Kanji" are typically from users interested in Japanese sites, the mutual information shows higher correlation with Japanese than with English or Chinese.

Table 5: Example of MI Scores

Query *n*-gram	URL *n*-gram	MI score
"kanji"	ja (japanese)	11.3862
"kanji"	zh (chinese)	6.2567
"kanji"	en (english)	4.2110

Table 6: Example of MI Score

Query *n*-gram	URL *n*-gram	MI score
"form"	pdf	4.9067
"form"	htm	1.0916
"video"	watch	5.7192
"video"	htm	-1.9079

Fourth, Table 6 shows the association between two query *n*-grams, "form" and "video", that at first glance may not actually look very informative for URL path selection. However, notice that the unigram "form" has a strong preference for pdf documents over more standard web pages with an html extension. Similarly, queries that include "video" convey a preference for URLs containing "watch", a characteristic URL *n*-gram for many video sharing websites.

It is reasonable to anticipate that incorporating such associations into a search engine's ranking function should help improve both search quality and user experience. Take the example where, there are two high ranking competing URLs for the query "irs 1040 form". Let's also assume both documents contain the same query relevant keywords, but one is an introduction of the "irs 1040 form" as an htm webpage and the other one is the real filing form given as a pdf document. Since in our dictionary, "form" is more associated with pdf than htm, we predict that most users would prefer the real pdf form directly, so it should be placed first in the list of query results. While click data for the exact query-url pairs confirms this preference, it is reassuring that we could identify it without needing to rely on seeing the specific query string before. As described in detail below, and motivated by this analysis, we designed our query-url click features based on the contents of the *n*-gram MI dictionary.

2.2 Query-URL *N*-gram Features

For our feature set, we explored the use of different query segmentation approaches (concept and affix segmentation) in order to increase the diversity of *n*-grams. In the following section, we use an unseen query "irs 1040 forms" and contrast it with the known query "irs 1040 form" from the last section.

2.2.1 Concept Segmentation Features

Query concept segmentation is a weighted query segmentation approach. Each query is analytically interpreted as being a main concept and a sub concept. We search for the unique segmentation of the query that maximizes its cumulative mutual information score with the URL *n*-grams. Main concepts and sub concepts are *n*-grams from the query that have the strongest association with URL *n*-grams and thus assist in identifying relevant landing URL *n*-grams when the whole query or the whole URL has not been seen.

Algorithm 1: Concept Segmentation

```
for U = domain, URL language, URL region, URL path do
  for j = 0... n-1 do
      M ⇐ W_{0...j}
      S ⇐ W_{j+1...n}
      for k = 0... m do
          curr_mi_M ⇐ arg max_{k=1...m} MI (M, U_k)
          curr_mi_S ⇐ arg max_{k=1...m} MI (S, U_k)
          if curr_mi_M + curr_mi_S > curr_best
          then
              curr_best = curr_mi_M + curr_mi_S
              mi_M ⇐ curr_mi_M
              mi_S ⇐ curr_mi_S
          end if
      end for
      adding mi_M as a feature
      adding mi_S as a feature
  end for
end for
```

Pseudo-code for generating query-url *n*-gram features based on the concept segmentation is given in Algorithm 1. Each query (Q) is composed of a number of words, w_1, w_2, w_3…,w_n. Each URL is segmented and categorized to four groups: domain, URL language, URL region and URL path. Each URL group has *m* number of URL *n*-grams. M is the main concept of Q and S is the sub concept of Q.

One potential drawback of such concept segmentation is data sparsity. When we look for the maximum of cumulative mutual information, we may obtain main concepts with very high mutual information and sub concepts which do not exist in the dictionary. In order to address this problem, we implement a second query segmentation method, affix segmentation, that is discussed in section 2.2.2.

Table 7 shows eight concept segmented features. "Coverage" is the percentage of query-url pairs that have valid feature values. Some of the samples do not have values because no clicks for the pairs were seen in the sample of data used to build the dictionary. When a pair does not have a value, the default value of zero is assigned. This default value is based on the assumption that unless we have evidence otherwise, we assume all query-url *n*-grams are statistically independent and thus provide no preference signal.

Table 7: Eight Features Generated based on Concept Segmentation.

Feature	Query *N*-gram	URL *N*-gram	Coverage (%)
MainDS	M	domain	54.09
SubDS	S	domain	30.46
MainLang	M	lang.	94.41
SubLang	S	lang.	72.40
MainReg	M	reg.	90.34
SubReg	S	reg.	68.19
MainPath	M	path	64.96
SubPath	S	path	58.76

Query-URL Domain Features are defined as the mutual information of a query *n*-gram and the domain level URL. There are two features in this category, one for the query main concept and one for the sub concept. They help to identify the user preferred host given a query.

Table 8: Example of Selecting Query Segmentation

MI(*q*,*u*)	irs.gov	
"irs"		11.2174
"1040"	11.6175	11.5550
"forms"	7.5049	
Cumulative MI	19.1224	22.7724
	Seg. 1	Seg.2

To illustrate the concept segmentation features, let's examine the query, **"**irs 1040 forms" in the context of the domain irs.gov. The query "irs 1040 forms" can be segmented either as "irs 1040" and "forms" or as "irs" and "1040 forms". As shown in Table 8, taking the cumulative maximum, the second segmentation scores higher than the first one. Therefore, the "irs" and "1040 forms" segmentation is preferred. The feature value for the main concept is 11.5550, and the sub concept is then assigned to be 11.2174.

Query-URL Language and Region Features are the mutual information of a query *n*-gram and URL language/region. They are used for providing language and region information.

Query-URL Path Features are the mutual information of a query *n*-gram and a URL path *n*-gram. While there are typically many URL path *n*-grams, only one URL path *n*-gram is selected to be paired with each query *n*-gram. The selected *n*-gram is the one that achieves the highest

cumulative maximum MI score. They are used for providing association between query *n*-grams and url *n*-grams such as "forms" and "pdf".

2.2.2 Affix Segmentation Features

As previously mentioned, affix segmentation addresses sparsity issues associated with concept segmentation. Here, we introduce the features generated based on affix segmentation. Pseudo-code for generating the features is given in Algorithm 2. Two query unigrams (w_0 and w_n) and one bigram (w_0w_n) is used. Each URL is segmented and categorized to four groups: domain, URL language, URL region and URL path. Each URL group has *m* number of URL *n*-grams.

This approach is complementary to the concept segmentation for long queries. The affix *n*-grams are in smaller unit, and therefore, are less sparse. In addition, the skip bigrams allow for generalizations using non-adjacent terms. Table 9 shows the coverage of the twelve affix features.

Algorithm 2: Affix Segmentation

```
for U = domain, URL language, URL region, URL path do
  for q = w0, wn, w0wn do
    for k = 0... m do
      curr_mi_q ⇐ arg max_{k=1...m} MI (q, U_k)
      if curr_mi_q > curr_best then
        curr_best = curr_mi_q
      end if
    end for
    adding curr_mi_q as a feature
  end for
end for
```

Table 9: Twelve Features Generated based on Affix Segmentation

Feature	Query *N*-gram	URL *N*-gram	Coverage (%)
PreDS	w_0	domain	48.09
SufDS	w_n	domain	47.72
PresufDS	w_0w_n	domain	23.57
PreLang	w_0	lang.	55.58
SufLang	w_n	lang.	58.22
PresufLang	w_0w_n	lang.	24.91
PreReg	w_0	reg.	93.82
SufReg	w_n	reg.	93.59
PresufReg	w_0w_n	reg.	69.29
PrePath	w_0	path	98.15
SufPath	w_n	path	97.80
PresufPath	w_0w_n	path	75.81

Query-url domain affix features has three features: MI(w_0, domain), MI(w_n, domain), and MI(w_0w_n, domain). In the example of "irs 1040 forms" and "irs.gov", the features are MI(irs, irs.gov), MI(forms, irs.gov), and MI(irs forms, irs.gov).

Query-url language and region affix features has three features respectively: MI(w_0, language), MI(w_n, language), MI(w_0w_n, language) MI(w_0, region), MI(w_n, region), and MI(w_0w_n, region). In the example of "irs 1040 forms", "en" and "us", the features are MI (irs, en), MI (forms, en), MI (irs forms, en), MI (irs, us), MI (forms, us), and MI (irs forms,us).

Query-url path affix features has three features: MI(w_0, path), MI(w_n, path), and MI(w_0w_n, path). In the example of "irs 1040 forms" and "www.irs.gov/pub/irs-pdf/f1040.pdf", there are four URL path *n*-grams, "pub", "irs", "pdf", and "f1040". The URL path *n*-gram, irs, gets maximum MI score. Therefore, the query-url path affix features are MI (irs, irs), MI (forms, irs), and MI (irs forms, irs).

We demonstrated the procedure to generate 20 query-url *n*-gram features, and in Section 3, we will present their effectiveness in relevance ranking.

3 Experiment

We evaluate the performance of query-url *n*-grams features (8 concept and 12 affix features) on a ranking application and analyze the results from several different perspectives.

3.1 Datasets

For all experiments, our training and test data are query-url pairs annotated with human judgments. In our data, we use five grades to evaluate relevance of a query and URL pair.

The data includes 94K queries for training and 3.4K queries for evaluation, and each query is associated with the top ranked URLs returned from a search engine. Totally, there are 916K query-url pairs for training and 42K pairs for testing. The queries are general and uniformly and randomly sampled with replacement, resulting in more frequent queries also appearing more frequently in our training and test sets.

3.2 Ranking Algorithm

GBRank is a supervised learning algorithm that uses boosted decision trees and incorporates the pair-wise information from the training data (Zheng et al, 2007). It is able to deal with a large amount of training data with hundreds of features. We use an internal C++ implementation of GBRank.

3.3 Evaluation Metric

We use Discounted Cumulative Gain (Järvelin and Kekäläinen, 2002) to evaluate our ranking accuracy. Discounted Cumulative Gain (DCG) has been widely used in evaluating the quality of search engine rankings and is defined as:

$$DCG_k = \sum_{i=1}^{k} \frac{G_i}{\log_2(i+1)} \quad (2)$$

G_i represents the editorial judgment of the i-th document. In this paper, we only report **normalized DCG$_5$**, which is an absolute DCG$_5$ normalized by a baseline, and **relative DCG$_5$ improvement**, which is an improvement normalized by the baseline. Note normalized DCG$_5$ is different than NDCG (Normalized Discounted Cumulative Gain defined in Järvelin and Kekäläinen, 2002). We use Wilcoxon signed test (Wilcoxon, 1945) to evaluate the significance for model comparison.

3.4 Feature Sets

Five feature sets are used in our experiments. Details are listed in Table 10.

Table 10: Five Feature Sets

Tag	Description
Base Feature Set	Core Feature Set and ExactM click features
Q-U *N*-gram Feature Set (I)	Base Feature Set and Q-U *N*-gram features
Core Feature Set	query-based, document-based, query-document based features
NavClick Feature Set	Core Feature Set and NavClick
Q-U *N*-gram Feature Set (II)	Core Feature Set and Q-U *N*-gram features

Base Feature Set is a strong baseline feature set from a state-of-the-art commercial search engine. This set includes NavClick features, and other internal ExactM click features. It is used for evaluating Query-URL *N*-gram Feature Set (I) in order to know whether query-url *n*-gram features can achieve gains when stacked on top of ExactM features.

Core Feature Set is a weaker variant of the baseline system that excludes ExactM click features. This system is used for evaluating NavClick Feature Set and Query-URL *N*-gram Feature Set (II) independently in order to study and contrast the effected queries.

3.5 Experimental Results

We compare the query-URL *N*-gram feature set (I) with the base feature set in Section 3.5.1, and contrast the NavClick features and the query-URL *N*-gram features (II) using the Core Feature Set in Section 3.5.2.

3.5.1 Query-URL *N*-gram Feature Set (I) versus Base Feature Set

As shown in Figure 2, Query-URL *N*-gram Feature Set (I) outperforms Base Feature Set. The additional 20 query-url *n*-gram features achieve statistically significant gains at a p-value < 0.0001 level, suggesting that they are complimentary to ExactM click features. Even though the query-url *n*-gram features are generated from the same data as the ExactM features, the gain is additive and stackable. The DCG$_5$ impact is 0.53% relative improvement when running GBRank using 2500 trees. Every data point is normalized by the DCG$_5$ of the baseline feature set using 2500 trees. This is represented in the graph as the rightmost point of Base Feature Set curve.

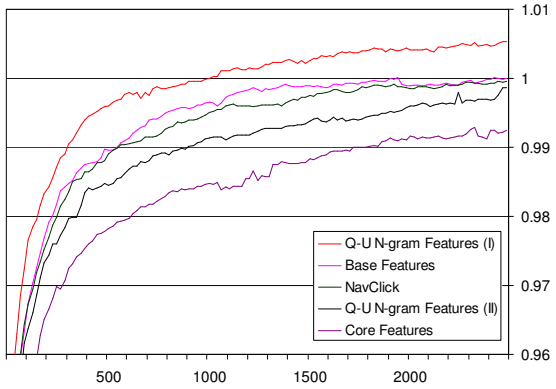


Figure 2: Comparison of the five feature sets on the normalized DCG$_5$ (Y-axis) against number of trees (X-axis).

3.5.2 NavClick and Query-URL *N*-gram Feature Set (II) versus Core Feature Set

We compare NavClick Feature Set and Query-URL *N*-gram Feature Set (II) in the context of Core Feature Set, in order to evaluate the two independently. As shown in Figure 2, both NavClick and Query-URL *N*-gram Feature Set (II) outperform Core Feature Set. It is not surprising that NavClick also outperforms Query-URL *N*-gram Feature Set (II) since the *n*-gram features are backoff of NavClick. However, their gains are competitive suggesting the query-url *n*-gram features are very good relevance indicators. The impact of NavClick and Query-URL *N*-gram Feature Set (II) is 0.72% and 0.62% relative DCG_5 improvement at Tree 2500 respectively.

3.5.3 Feature Importance

Using the GBRank model, features are evaluated and sequentially selected to build the boosted decision trees. The split of each node increases the DCG during training. We evaluate a feature's importance by aggregating the DCG impact of the feature over all trees (Zheng et al., 2007). Here, the feature importance is rescaled so that the feature with largest DCG impact is assigned a normalized score of 1. Figure 3 illustrates the relative influence of each of query-url *n*-gram feature. Of these, *n*-gram features associated with a domain name (i.e., MainDS) rank highest.

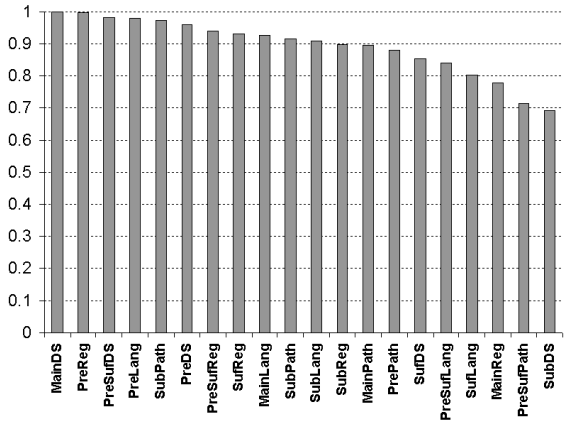


Figure 3: Feature importance of query-url *n*-gram features. The importance (Y axis) is normalized so that the most important feature (MainDS)'s importance is 1.

3.6 Analysis

We access system performance with respect to both query length and frequency using the two click features sets in combination with the Core Feature Set in order to gain insight into the effected queries.

3.6.1 Query Length

As shown in Table 11, NavClick (NavClick Feature Set) best improves relevance for two word queries. In contrast, Query-url *n*-gram features in isolation (Query-URL *N*-gram Feature II) are able to show sizable improvements on longer queries, while slightly degrading performance on short 1-word queries. Using both feature sets together (Query-URL *N*-gram Feature I) results in improvement for queries of all lengths.

These results suggest that the strong signal being provided by NavClick for short queries helps to compensate for any additional noisy introduced by the *n*-gram features, while allowing the *n*-gram features to handle longer queries that are less well covered by NavClick. These longer queries are exactly the type of queries our query-url *n*-gram features were designed to help with.

Table 11: Relative DCG_5 Improvement of NavClick, Query-URL *N*-gram (II), and Query-URL *N*-gram Features (I) vs Core Feature Set

Length	NavClick vs Core (%)	QU *N*-gram (II) vs Core (%)	QU *N*-gram (I) vs Core (%)
1 word	0.03	-0.04	0.62
2 words	1.04	1.06	1.58
3 words	1.00	1.44	2.12
4+ words	0.4	0.68	1.01

3.6.2 Query Frequency

We found that query-url *n*-gram features improve tail queries. Head queries are considered as top two million frequent queries in our traffic and tail queries include anything outside of that range.

Table 12: Relative DCG_5 Improvement of NavClick, Query-URL *N*-gram Features (II) and Query-URL *N*-gram Features (I) vs Core Feature Set

	NavClick vs Core (%)	QU *N*-gram (II) vs Core (%)	QU *N*-gram (I) vs Core (%)
Head	0.91	-0.15	1.11
Tail	0.59	1.11	1.40

As shown in Table 12, query-url *n*-gram features (Query-URL Feature Set II) differ from NavClick (NavClick Feature Set) in that they get

more gain from tail queries. Together, they (Query-URL Feature Set I) improve both head and tail queries.

3.7 Case Study

Below we examine queries from the test set and analyze the effects of Query-URL *N*-gram Feature Set (II) versus Core Feature Set.

3.7.1 Positive Cases

1) Animal shelter in va: this query targets a specific geographic location. Using the baseline feature set, the root url wvanimalshelter.org is incorrectly ranked higher than www.netpets.com/cats/catresc/virginia.htm. Without any additionally ranking information, general URLs (root) tend to be ranked more highly than more specific URLs (path), as the root pages tend to be more popular. However, our new features express a preference between "va" and "virginia", and this correctly flips the ranking order.
2) Myspace profile generator: www. myspacgens.com/handler.php?gen=profile was incorrectly ranked higher than www.profilemods.com/myspace-generators. Our new features convey a high user preference association between "profile generator" and the domain profilemods.com, which helps to correctly swap the order.

3.7.2 Negative Cases

We determined that negative cases where the baseline feature set outperforms the new features are typically one word navigational queries such as "craigslist". However, after we combine the query-url *n*-gram features with NavClick, one word navigational queries are ranked correctly.

4 Related Work

Our work is mainly related to Gao et al. (2009) and Bilenko and White (2008). Gao et al. (2009) addressed the sparsity issue by propagating click information among similar queries in the same cluster. Their idea is based on an observation that similar queries go to similar pages. When two queries have similar clicked URLs, it is likely that they share clicked URLs. In contrast, our idea is to utilize NLP techniques to break down long, infrequent queries into shorter, frequent queries. The two approaches can be mutually beneficial. Bilenko and White (2008) expanded click data with a search engine by using post-search user experience collected from toolbars. Toolbars keep track of users' click behavior both when they are using the search engine directly and beyond. Their relevance features are built based on whole session clicks extracted from the toolbar. In contrast, our *n*-gram features are built on search engine clicks directly. We should be able to expand our method to integrate the post-search clicks with toolbar data.

Other related work can be found in the domain of query rewriting. Our *n*-gram dictionary was originally designed for query rewriting. Query rewriting (Xu and Croft, 1996; Salton and Voorhees, 1984) reformulates a query to its synonyms or related terms automatically. However, the coverage of query rewriting is normally small, because an inappropriate rewrite can cause significant decrease in precision. In contrast, our approach can cover a larger number of queries without decreasing precision, because it does not need to make a binary decision whether a query should be reformulated. The association scores between queries and rewrites are used as ranking features which are trained discriminatively toward search quality.

5 Conclusion

In this paper, we presented a set of straightforward yet informative query-url *n*-gram features. They allow for generalization of limited user click data to large amounts of unseen query-url pairs. Our experiments showed such features gave significant improvement over models without using the features. In addition, we mined an interesting dictionary which contains informative, but not necessarily obvious, query-url synonym pairs such as "form" and "pdf". We are currently extending our work to a variety of exact match features and different sources of clickthrough logs.

Acknowledgement

Thanks to the anonymous reviewers for detailed suggestion and our colleagues: Jon Degenhardt and Narayanan Sadagopan for assistance on generating clickthrough data, Jiang Chen for developing the decision tree package, Xiangyu Jin for a discussion on map/reduce, Benoît Dumoulin, Fuchun Peng, Yumao Lu, and Xing Wei for productizing the work, and Rosie Jones, Su-lin Wu, Bo Long, Xin Li and Ruiqiang Zhang for comments on an earlier draft.

References

Agichtein, E., E. Brill, and S. Dumais. 2006. Improving web search ranking by incorporating user behavior information. In Proceedings of the ACM SIGIR 29.

Agichtein, Eugene, Zijian Zheng. 2006. Identifying "best bet" web search results by mining past user behavior. In Proceedings of KDD.

Bilenko, Mikhail and Ryen W. White. 2008. Mining the search trails of surfing crowds: identifying relevant websites from user activity. In Proceedings of WWW.

Brants, T. 2000. Tnt: a statistical part-ofspeech tagger. In Proceedings of ANLP 6.

Craswell, Nick and Martin Szummer. 2007. Random walks on the click graph. In Proceedings of SIGIR.

Craswell, Nick, Onno Zoeter, Michael Taylor, Bill Ramsey. 2008. An experimental comparison of click position-bias models in WSDM.

Dupret, Georges, Benjamin Piwowarski. 2008. A user browsing model to predict search engine click data from past observations. In Proceedings of SIGIR 31.

Gale, William A. and Kenneth W. Church. 1991. Identifying word correspondence in parallel texts. In Proceedings of HLT 91.

Gao, Jianfeng, Wei Yuan, Xiao Li, Kefeng Deng, and Jian-Yun Nie. 2009. Smoothing Clickthrough Data for Web Search Ranking. In Proceedings of SIGIR 32.

Järvelin, K. and J. Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques, Journal ACM Transactions on Information Systems, 20: 422-446.

Klein, D. and C. Manning. 2003. Accurate unlexicalized parsing. In Proceedings of ACL 41.

Lin, Chin-Yew and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram. In In Proceedings of ACL 42.

Lu, Yumao, Fuchun Peng, Xin Li and Nawaaz Ahmed, 2006, Coupling Feature Selection and Machine Learning Methods for Navigational Query Identification, In Proceeding of CIKM.

Radlinski, F., Kurup, M. and Joachims, T. 2007. Active exploration for learning rankings from clickthrough data. In *SIGKDD*.

Salton G. and E. Voorhees. 1984. Comparison of two methods for Boolean query relevancy feedback. Information Processing & Management, 20(5).

Wilcoxon, F. 1945. Individual Comparisons by Ranking Methods. Biometrics, 1:80–83.

Xu Q. and W. Croft. 1996. Query expansion using local and global document analysis. In Proceed of the 19th annual international ACM SIGIR.

Zheng, Z., H. Zha, K. Chen, and G. Sun. 2007. A regression framework for learning ranking functions using relative relevance judgments. In Proceedings of SIGIR 30.

The role of named entities in Web People Search

Javier Artiles
UNED NLP & IR group
Madrid, Spain
javart@bec.uned.es

Enrique Amigó
UNED NLP & IR group
Madrid, Spain
enrique@lsi.uned.es

Julio Gonzalo
UNED NLP & IR group
Madrid, Spain
julio@lsi.uned.es

Abstract

The ambiguity of person names in the Web has become a new area of interest for NLP researchers. This challenging problem has been formulated as the task of clustering Web search results (returned in response to a person name query) according to the individual they mention. In this paper we compare the coverage, reliability and independence of a number of features that are potential information sources for this clustering task, paying special attention to the role of named entities in the texts to be clustered. Although named entities are used in most approaches, our results show that, independently of the Machine Learning or Clustering algorithm used, named entity recognition and classification per se only make a small contribution to solve the problem.

1 Introduction

Searching the Web for names of people is a highly ambiguous task, because a single name tends to be shared by many people. This ambiguity has recently become an active research topic and, simultaneously, in a relevant application domain for web search services: Zoominfo.com, Spock.com, 123people.com are examples of sites which perform web people search, although with limited disambiguation capabilities.

A study of the query log of the AllTheWeb and Altavista search sites gives an idea of the relevance of the people search task: 11-17% of the queries were composed of a person name with additional terms and 4% were identified as person names (Spink et al., 2004). According to the data available from 1990 U.S. Census Bureau, only 90,000 different names are shared by 100 million people (Artiles et al., 2005). As the amount of information in the WWW grows, more of these people are mentioned in different web pages. Therefore, a query for a common name in the Web will usually produce a list of results where different people are mentioned.

This situation leaves to the user the task of finding the pages relevant to the particular person he is interested in. The user might refine the original query with additional terms, but this risks excluding relevant documents in the process. In some cases, the existence of a predominant person (such as a celebrity or a historical figure) makes it likely to dominate the ranking of search results, complicating the task of finding information about other people sharing her name. The Web People Search task, as defined in the first WePS evaluation campaign (Artiles et al., 2007), consists of grouping search results for a given name according to the different people that share it.

Our goal in this paper is to study which document features can contribute to this task, and in particular to find out which is the role that can be played by named entities (NEs): (i) How reliable is NEs overlap between documents as a source of evidence to cluster pages? (ii) How much recall does it provide? (iii) How unique is this signal? (i.e. is it redundant with other sources of information such as n-gram overlap?); and (iv) How sensitive is this signal to the peculiarities of a given NE recognition system, such as the granularity of its NE classification and the quality of its results?

Our aim is to reach conclusions which are are not tied to a particular choice of Clustering or Machine Learning algorithms. We have taken two decisions in this direction: first, we have focused on the problem of deciding whether two web pages refer to the same individual or not (page coreference task). This is the kind of relatedness measure that most clustering algorithms use, but in this way we can factor out the algorithm and its parameter settings. Second, we have developed a measure, *Maximal Pairwise Accuracy* (PWA) which, given

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 534–542,
Singapore, 6-7 August 2009.

an information source for the problem, estimates an upper bound for the performance of any Machine Learning algorithm using this information. We have used PWA as the basic metric to study the role of different document features in solving the coreference problem, and then we have checked the predictive power of PWA with a Decision Tree algorithm.

The remainder of the paper is organised as follows. First, we examine the previous work in Section 2. Then we describe the our experimental settings (datasets and features we have used) in Section 3 and our empirical study in Section 4. The paper ends with some conclusions in Section 5.

2 Previous work

In this section we will discuss (i) the state of the art in Web People Search in general, focusing on which features are used to solve the problem; and (ii) lessons learnt from the WePS evaluation campaign where most approaches to the problem have been tested and compared.

The disambiguation of person names in Web results is usually compared to two other Natural Language Processing tasks: Word Sense Disambiguation (WSD) (Agirre and Edmonds, 2006) and Cross-document Coreference (CDC) (Bagga and Baldwin, 1998). Most of early research work on person name ambiguity focuses on the CDC problem or uses methods found in the WSD literature. It is only recently that the web name ambiguity has been approached as a separate problem and defined as an NLP task - *Web People Search* - on its own (Artiles et al., 2005; Artiles et al., 2007).

Therefore, it is useful to point out some crucial differences between WSD, CRC and WePS:

- WSD typically concentrates in the disambiguation of common words (nouns, verbs, adjectives) for which a relatively small number of senses exist, compared to the hundreds or thousands of people that can share the same name.
- WSD can rely on dictionaries to define the number of possible senses for a word. In the case of name ambiguity no such dictionary is available, even though in theory there is an exact number of people that can be accounted as sharing the same name.
- The objective of CDC is to reconstruct the coreference chain for every mention of a person. In Web person name disambiguation it suffices to group the documents that contain at least one mention to the same person.

Before the first WePS evaluation campaign in 2007 (Artiles et al., 2007), research on the topic was not based on a consistent task definition, and it lacked a standard manually annotated testbed. In the WePS task, systems were given the top web search results produced by a person name query. The expected output was a clustering of these results, where each cluster should contain all and only those documents referring to the same individual.

2.1 Features for Web People Search

Many different features have been used to represent documents where an ambiguous name is mentioned. The most basic is a **Bag of Words** (BoW) representation of the document text. Within-document coreference resolution has been applied to produce summaries of text surrounding occurrences of the name (Bagga and Baldwin, 1998; Gooi and Allan, 2004). Nevertheless, the full document text is present in most systems, sometimes as the only feature (Sugiyama and Okumura, 2007) and sometimes in combination with others - see for instance (Chen and Martin, 2007; Popescu and Magnini, 2007)-. Other representations use the link structure (Malin, 2005) or generate graph representations of the extracted features (Kalashnikov et al., 2007).

Some researchers (Cucerzan, 2007; Nguyen and Cao, 2008) have explored the use of **Wikipedia information** to improve the disambiguation process. Wikipedia provides candidate entities that are linked to specific mentions in a text. The obvious limitation of this approach is that only celebrities and historical figures can be identified in this way. These approaches are yet to be applied to the specific task of grouping search results.

Biographical features are strongly related to NEs and have also been proposed for this task due to its high precision. Mann (2003) extracted these features using lexical patterns to group pages about the same person. Al-Kamha (2004) used a simpler approach, based on hand coded features (e.g. email, zip codes, addresses, etc). In Wan (2005), biographical information (person name, title, organisation, email address and phone number) improves the clustering results when combined with lexical features (words from the doc-

ument) and NE (person, location, organisation).

The most used feature for the Web People Search task, however, are **NEs**. Ravin (1999) introduced a rule-based approach that tackles both variation and ambiguity analysing the structure of names. In most recent research, NEs (person, location and organisations) are extracted from the text and used as a source of evidence to calculate the similarity between documents -see for instance (Blume, 2005; Chen and Martin, 2007; Popescu and Magnini, 2007; Kalashnikov et al., 2007)-. For instance, Blume (2005) uses NEs coocurring with the ambiguous mentions of a name as a key feature for the disambiguation process. Saggion (2008) compared the performace of NEs versus BoW features. In his experiments a only a representation based on Organisation NEs outperformed the word based approach. Furthermore, this result is highly dependent on the choice of metric weighting (NEs achieve high precision at the cost of a low recall and viceversa for BoW).

In summary, the most common document representations for the problem include BoW and NEs, and in some cases biographical features extracted from the text.

2.2 Named entities in the WePS campaign

Among the 16 teams that submitted results for the first WePS campaign, 10 of them[1] used NEs in their document representation. This makes NEs the second most common type of feature; only the BoW feature was more popular. Other features used by the systems include noun phrases (Chen and Martin, 2007), word n-grams (Popescu and Magnini, 2007), emails and URLs (del Valle-Agudo et al., 2007), etc. In 2009, the second WePS campaign showed similar trends regarding the use of NE features (Artiles et al., 2009).

Due to the complexity of systems, the results of the WePS evaluation do not provide a direct answer regarding the advantages of using NEs over other computationally lighter features such as BoW or word n-grams. But the WePS campaigns did provide a useful, standardised resource to perform the type of studies that were not possible before. In the next Section we describe this dataset and how it has been adapted for our purposes.

[1]By team ID: CU-COMSEM, IRST-BP, PSNUS, SHEF, FICO, UNN, AUG, JHU1, DFKI2, UC3M13

3 Experimental settings

3.1 Data

We have used the testbeds from WePS-1 (Artiles et al., 2007)[2] and WePS-2 (Artiles et al., 2009) evaluation campaigns [3].

Each WePS dataset consists of 30 test cases: a random sample of 10 names from the US Census, 10 names from Wikipedia, and 10 names from Programme Committees in the Computer Science domain (ACL and ECDL). Each test case consists of, at most, 100 web pages from the top search results of a web search engine, using a (quoted) person name as query.

For each test case, annotators were asked to organise the web pages in groups where all documents refer to the same person. In cases where a web page refers to more than one person using the same ambiguous name (e.g. a web page with search results from Amazon), the document is assigned to as many groups as necessary. Documents were discarded when they did not contain any useful information about the person being referred.

Both the WePS-1 and WePS-2 testbeds have been used to evaluate clustering systems by WePS task participants, and are now the standard testbed to test Web People Search systems.

3.2 Features

The evaluated features can be grouped in four main groups: token-based, n-grams, phrases and NEs. Wherever possible, we have generated *local* versions of these features that only consider the sentences of the text that mention the ambiguous person name[4]. Token-based features considered include document full text tokens, lemmas (using the OAK analyser, see below), title, snippet (returned in the list of search results) and URL (tokenised using non alphanumeric characters as boundaries) tokens. English stopwords were removed, including Web specific stopwords, as file and domain extensions, etc.

We generated **word n-grams** of length 2 to 5,

[2]The WePS-1 corpus includes data from the Web03 testbed (Mann, 2006) which follows similar annotation guidelines, although the number of document per ambiguous name is more variable.

[3]Both corpora are available from the WePS website http://nlp.uned.es/weps

[4]A very sparse feature might never occur in a sentence with the person name. In that cases there is no *local* version of the feature.

using the sentences found in the document text. Punctuation tokens (commas, dots, etc) were generalised as the same token. N-grams were discarded when they were composed only of stopwords or when they did not contain at least one token formed by alphanumeric characters (e.g. n-grams like "at the" or "# @"). **Noun phrases** (using OAK analyser) were detected in the document and filtered in a similar way.

Named entities were extracted using two different tools: the Stanford NE Recogniser and the OAK System[5].

Stanford NE Recogniser[6] is a high-performance Named Entity Recognition (NER) system based on Machine Learning. It provides a general implementation of linear chain Conditional Random Field sequence models and includes a model trained on data from CoNLL, MUC6, MUC7, and ACE newswire. Three types of entities were extracted: person, location and organisation.

OAK[7] is a rule based English analyser that includes many functionalities (POS tagger, stemmer, chunker, Named Entity (NE) tagger, dependency analyser, parser, etc). It provides a fine grained NE recognition covering 100 different NE types (Sekine, 2008). Given the sparseness of most of these fine-grained NE types, we have merged them in coarser groups: event, facility, location, person, organisation, product, periodx, timex and numex.

We have also used the results of a **baseline NE** recognition for comparison purposes. This method detects sequences of two or more uppercased tokens in the text, and discards those that are found lowercased in the same document or that are composed solely of stopwords.

Other features are: emails, outgoing links found in the web pages and two boolean flags that indicate whether a pair of documents is linked or belongs to the same domain. Because of their low impact in the results these features haven't received an individual analysis, but they are included in the "all features" combination in Figure 7.

[5]From the output of both systems we have discarded person NEs made of only one token (these are often first names that significantly deteriorate the quality of the comparison between documents).

[6]http://nlp.stanford.edu/software/CRF-NER.shtml

[7]http://nlp.cs.nyu.edu/oak . OAK was also used to detect noun phrases and extract lemmas from the text.

4 Experiments and results

4.1 Reformulating WePS as a classification task

As our goal is to study the impact of different features (information sources) in the task, a direct evaluation in terms of clustering has serious disadvantages. Given the output of a clustering system it is not straightforward to assess why a document has been assigned to a particular cluster. There are at least three different factors: the document similarity function, the clustering algorithm and its parameter settings. Features are part of the document similarity function, but its performance in the clustering task depends on the other factors as well. This makes it difficult to perform error analysis in terms of the features used to represent the documents.

Therefore we have decided to transform the clustering problem into a classification problem: deciding whether two documents refer to the same person. Each pair of documents in a name dataset is considered a classification instance. Instances are labelled as coreferent (if they share the same cluster in the gold standard) or non coreferent (if they do not share the same cluster). Then we can evaluate the performance of each feature separately by measuring its ability to rank coreferent pairs higher and non coreferent pairs lower. In the case of feature combinations we can study them by training a classifier or using the maximal pairwise accuracy methods (explained in Section 4.3).

Each instance (pair of documents) is represented by the similarity scores obtained using different features and similarity metrics. We have calculated for each feature three similarity metrics: Dice's coefficient, cosine (using standard tf.idf weighting) and a measure that simply counts the size of the intersection set for a given feature between both documents. After testing these metrics we found that Dice provides the best results across different feature types. Differences between Dice and cosine were consistent, although they were not especially large. A possible explanation is that Dice does not take into account the redundancy of an n-gram or NE in the document, and the cosine distance does. This can be a crucial factor, for instance, in the document retrieval by topic; but it doesn't seem to be the case when dealing with name ambiguity.

The resulting classification testbed consists of 293,914 instances with the distribution shown in

Table 1, where each instance is represented by 69 features.

	true	false	total
WePS1	61,290	122,437	183,727
WePS2	54,641	55,546	110,187
WePS1+WePS2	115,931	177,983	293,914

Table 1: Distribution of classification instances

4.2 Analysis of individual features

There are two main aspects related with the usefulness of a feature for WePS task. The first one is its performance. That is, to what extent the similarity between two documents according to a feature implies that both mention the same person. The second aspect is to what extent a feature is orthogonal or redundant with respect to the standard token based similarity.

4.2.1 Feature performance

According to the transformation of WePS clustering problem into a classification task (described in Section 4.1), we follow the next steps to study the performance of individual features. First, we compute the Dice coefficient similarity over each feature for all document pairs. Then we rank the document pair instances according to these similarities. A good feature should rank positive instances on top. If the number of coreferent pairs in the top n pairs is t_n and the total number of coreferent pairs is t, then $P = \frac{t_n}{n}$ and $R = \frac{t_n}{t}$. We plot the obtained precision/recall curves in Figures 1, 2, 3 and 4.

From the figures we can draw the following conclusions:

First, considering subsets of tokens or lemmatised tokens does not outperform the basic token distance (figure 1 compares token-based features). We see that only local and snippet tokens perform slightly better at low recall values, but do not go beyond recall 0.3.

Second, shallow parsing or n-grams longer than 2 do not seem to be effective, but using bi-grams improves the results in comparison with tokens. Figure 2 compares n-grams of different sizes with noun phrases and tokens. Overall, noun phrases have a poor performance, and bi-grams give the best results up to recall 0.7. Four-grams give slightly better precision but only reach 0.3 recall, and three-grams do not give better precision than bi-grams.

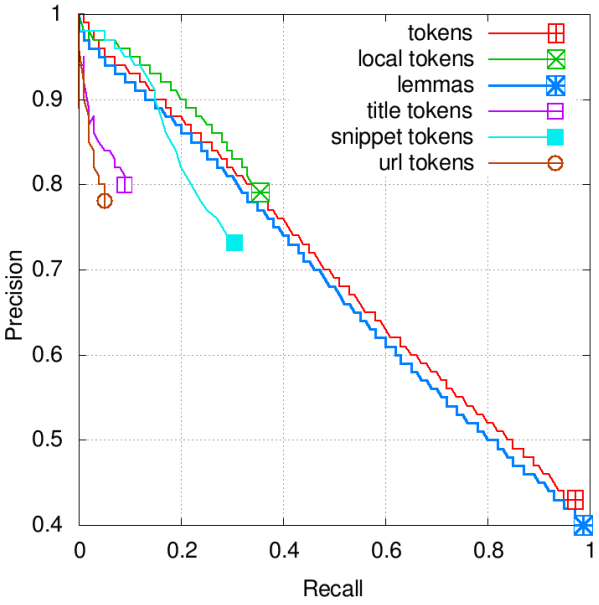


Figure 1: Precision/Recall curve of token-based features

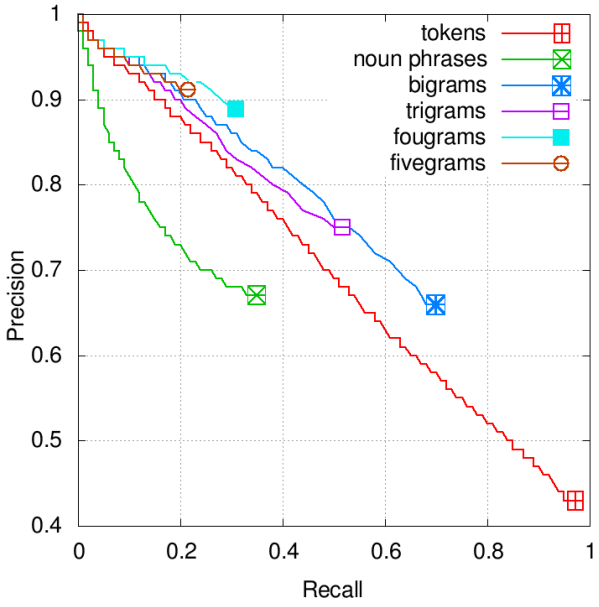


Figure 2: Precision/Recall curve of word n-grams

Third, individual types of NEs do not improve over tokens. Figure 3 and Figure 4 display the results obtained by the Stanford and OAK NER tools respectively. In the best case, Stanford person and organisation named entities obtain results that match the tokens feature, but only at lower levels of recall.

Finally, using different NER systems clearly leads to different results. Surprisingly, the baseline NE system yields better results in a one to one comparison, although it must be noted that this baseline agglomerates different types of en-

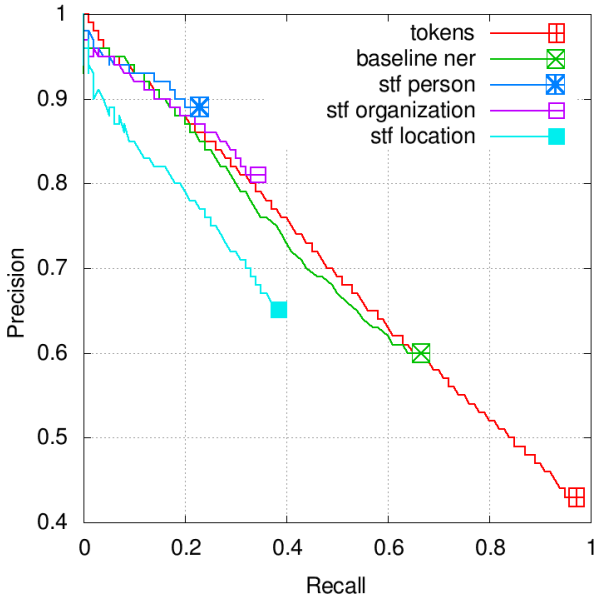


Figure 3: Precision/Recall curve of NEs obtained with the Stanford NER tool

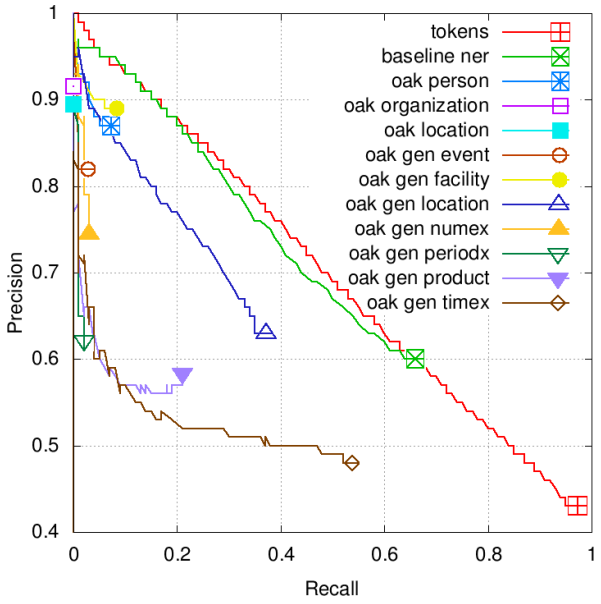


Figure 4: Precision/Recall curve of NEs obtained with the OAK NER tool

tities that are separated in the case of Stanford and OAK, and this has a direct impact on its recall. The OAK results are below the tokens and NE baseline, possibly due to the sparseness of its very fine grained features. In NE types, cases such as person and organisation results are still lower than obtained with Stanford.

4.2.2 Redundancy

In addition to performance, named entities (as well as other features) are potentially useful for the task only if they provide information that complements (i.e. that does not substantially overlap) the basic token based metric. To estimate this redundancy, let us consider all document tuples of size four $< a, b, c, d >$. In 99% of the cases, token similarity is different for $< a, b >$ than for $< c, d >$. We take combinations such that $< a, b >$ are more similar to each other than $< c, d >$ according to tokens. That is:

$$\text{sim}_{\text{token}}(a, b) > \text{sim}_{\text{token}}(c, d)$$

Then for any other feature similarity $sim_x(a, b)$, we will talk about *redundant* samples when $sim_x(a, b) > sim_x(c, d)$, *non redundant* samples when $sim_x(a, b) < sim_x(c, d)$, and *non informative* samples when $sim_x(a, b) = sim_x(c, d)$. If all samples are redundant or non informative, then sim_x does not provide additional information for the classification task.

Figure 5 shows the proportion of redundant, non redundant and non informative samples for several similarity criteria, as compared to token-based similarity. In most cases NE based similarities give little additional information: the baseline NE recogniser, which has the largest independent contribution, gives additional information in less than 20% of cases.

In summary, analysing individual features, the NEs do not outperform BoW in terms of the classification task. In addition, NEs tend to be redundant regarding BoW. However, if we are able to combine optimally the contributions of the different features, the BoW approach could be improved. We address this issue in the next section.

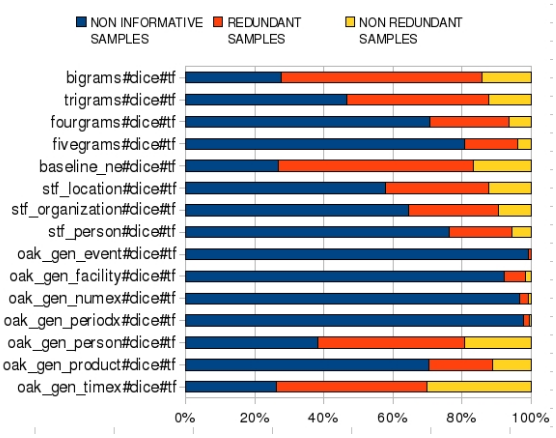


Figure 5: Independence of similarity criteria with respect to the token based feature

4.3 Analysis of feature combinations

Up to now we have analysed the usefulness of individual features for the WePS Task. However, this begs to ask to what extent the NE features can contribute to the task when they are combined together and with token and n-gram based features. First, we use each feature combinations as the input for a Machine Learning algorithm. In particular, we use a Decision Tree algorithm and WePS-1 data for training and WePS-2 data for testing. The Decision Tree algorithm was chosen because we have a small set of features to train (similarity metrics) and some of these features output Boolean values.

Results obtained with this setup, however, can be dependent on the choice of the ML approach. To overcome this problem, in addition to the results of a Decision Tree Machine Learning algorithm, we introduce a *Maximal Pairwise Accuracy* (MPA) measure that provides an upper bound for any machine learning algorithm using a feature combination.

We can estimate the performance of an individual similarity feature x in terms of accuracy. It is considered a correct answer when the similarity $x(a, a')$ between two pages referring to the same person is higher than the similarity $x(b, c)$ between two pages referring to different people. Let us call this estimation *Pairwise Accuracy*. In terms of probability it can be defined as:

$$PWA = \text{Prob}(x(a, a') > x(c, d))$$

PWA is defined over a single feature (similarity metric). When considering more than one similarity measure, the results depend on how measures are weighted. In that case we assume that the best possible weighting is applied. When combining a set of features $X = \{x_1 \dots x_n\}$, a perfect Machine Learning algorithm would learn to always "listen" to the features giving correct information and ignore the features giving erroneous information. In other words, if at least one feature gives correct information, then the perfect algorithm would produce a correct output. This is what we call the *Maximal Pairwise Accuracy* estimation of an upper bound for any ML system using the set of features X:

$$\text{MaxPWA}(X) =$$
$$\text{Prob}(\exists x \in X . x(a, a') > x(c, d))$$

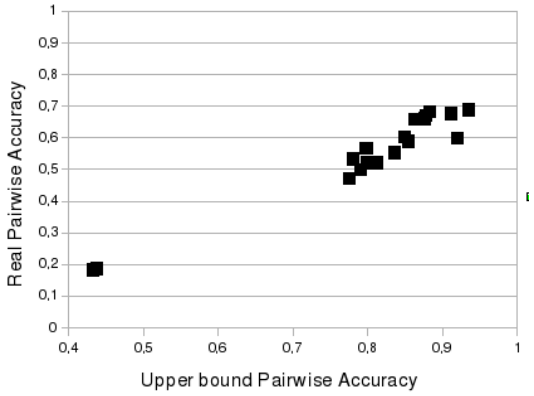


Figure 6: Estimated PWA upper bound versus the real PWA of decision trees trained with feature combinations

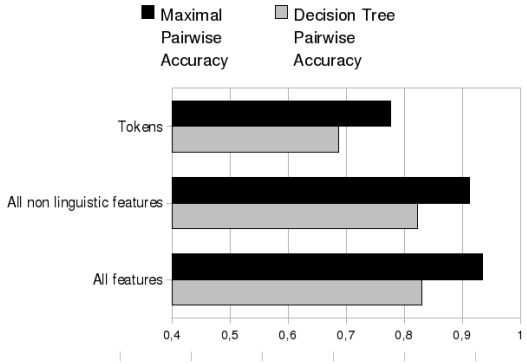


Figure 7: Maximal Pairwise Accuracy vs. results of a Decision Tree

The upper bound (MaxPWA) of feature combinations happens to be highly correlated with the PWA obtained by the Decision Tree algorithm (using its confidence values as a similarity metric). Figure 6 shows this correlation for several features combinations. This is an indication that the Decision Tree is effectively using the information in the feature set.

Figure 7 shows the PWA upper bound estimation and the actual PWA performance of a Decision Tree ML algorithm for three combinations: (i) all features; (ii) non linguistic features, i.e., features which can be extracted without natural language processing machinery: tokens, url, title, snippet, local tokens, n-grams and local n-grams; and (iii) just tokens. The results show that according to both the Decision Tree results and the upper-bound (MaxPWA), adding new features to tokens improves the classification. However, taking non-linguistic features obtains similar results than taking all features. Our conclusion is that NE features are useful for the task, but do not seem to offer a

competitive advantage when compared with non-linguistic features, and are more computationally expensive. Note that we are using NE features in a direct way: our results do not exclude the possibility of effectively exploiting NEs in more sophisticated ways, such as, for instance, exploiting the underlying social network relationships between NEs in the texts.

4.3.1 Results on the clustering task

In order to validate our results, we have tested whether the classifiers learned with our feature sets lead to competitive systems for the full clustering task. In order to do so, we use the output of the classifiers as similarity metrics for a particular clustering algorithm, using WePS-1 to train the classifiers and WePS-2 for testing.

We have used a Hierarchical Agglomerative Clustering algorithm (HAC) with single linkage, using the classifier's confidence value in the negative answer for each instance as a distance metric[8] between document pairs. HAC is the algorithm used by some of the best performing systems in the WePS-2 evaluation. The distance threshold was trained using the WePS-1 data. We report results with the official WePS-2 evaluation metrics: extended B-Cubed Precision and Recall (Amigó et al., 2008).

Two Decision Tree models were evaluated: (i) *ML-ALL* is a model trained using all the available features (which obtains 0.76 accuracy in the classification task) (ii) *ML-NON_LING* was trained with all the features except for OAK and Stanford NEs, noun phrases, lemmas and gazetteer features (which obtains 0.75 accuracy in the classification task). These are the same classifiers considered in Figure 7.

Table 2 shows the results obtained in the clustering task by the two DT models, together with the four top scoring WePS-2 systems and the average values for all WePS-2 systems. We found that a ML based clustering using only non linguistic information slightly outperforms the best participant in WePS-2. Surprisingly, adding linguistic information (NEs, noun phrases, etc.) has a small negative impact on the results (0.81 versus 0.83), although the classifier with linguistic information was a bit better than the non-linguistic one. This seems to be another indication that the use of noun phrases and other linguistic features to improve the task is non-obvious to say the least.

		B-Cubed	
run	F-α $=_{0.5}$	Pre.	Rec.
ML-NON_LING	.83	.91	.77
S-1	.82	.87	.79
ML- ALL	.81	.89	.76
S-2	.81	.85	.80
S-3	.81	.93	.73
S-4	.72	.82	.66
WePS-2 systems aver.	.61	.74	.63

Table 2: Evaluation on the WePS-2 clustering task

5 Conclusions

We have presented an empirical study that tries to determine the potential role of several sources of information to solve the Web People Search clustering problem, with a particular focus on studying the role of named entities in the task.

To abstract the study from the particular choice of a clustering algorithm and a parameter setting, we have reformulated the problem as a coreference classification task: deciding whether two pages refer to the same person or not. We have also proposed the *Maximal Pairwise Accuracy* estimation that establish an upper bound for the results obtained by any Machine Learning algorithm using a particular set of features.

Our results indicate that (i) NEs do not provide a substantial competitive advantage in the clustering process when compared to a rich combination of simpler features that do not require linguistic processing (local, global and snippet tokens, n-grams, etc.); (ii) results are sensitive to the NER system used: when using all NE features for training, the richer number of features provided by OAK seems to have an advantage over the simpler types in Stanford NER and the baseline NER system.

This is not exactly a prescription against the use of NEs for Web People Search, because linguistic knowledge can be useful for other aspects of the problem, such as visualisation of results and description of the persons/clusters obtained: for example, from a user point of view a network of the connections of a person with other persons and organisations (which can only be done with NER) can be part of a person's profile and may help as a summary of the cluster contents. But from the perspective of the clustering problem per se, a direct use of NEs and other linguistic features does not seem to pay off.

[8] The DT classifier output consists of two confidence values, one for the positive and one for the negative answer, that add up to 1.0 .

Acknowledgments

This work has been partially supported by the Regional Government of Madrid, project MAVIR S0505-TIC0267.

References

Eneko Agirre and Philip Edmonds, editors. 2006. *Word Sense Disambiguation: Algorithms and Applications*. Springer.

Reema Al-Kamha and David W. Embley. 2004. Grouping search-engine returned citations for person-name queries. In *WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management*. ACM Press.

Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2008. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*.

Javier Artiles, Julio Gonzalo, and Felisa Verdejo. 2005. A testbed for people searching strategies in the www. In *SIGIR*.

Javier Artiles, Julio Gonzalo, and Satoshi Sekine. 2007. The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*. ACL.

Javier Artiles, Julio Gonzalo, and Satoshi Sekine. 2009. Weps 2 evaluation campaign: overview of the web people search clustering task. In *WePS 2 Evaluation Workshop. WWW Conference 2009*.

Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th international conference on Computational linguistics*. ACL.

Matthias Blume. 2005. Automatic entity disambiguation: Benefits to ner, relation extraction, link analysis, and inference. In *International Conference on Intelligence Analysis*.

Ying Chen and James H. Martin. 2007. Cu-comsem: Exploring rich features for unsupervised web personal name disambiguation. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*. ACL.

Silviu Cucerzan. 2007. Large scale named entity disambiguation based on wikipedia data. In *The EMNLP-CoNLL-2007*.

David del Valle-Agudo, César de Pablo-Sánchez, and María Teresa Vicente-Díez. 2007. Uc3m-13: Disambiguation of person names based on the composition of simple bags of typed terms. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*. ACL.

Chung Heong Gooi and James Allan. 2004. Cross-document coreference on a large scale corpus. In *HLT-NAACL*.

Dmitri V. Kalashnikov, Stella Chen, Rabia Nuray, Sharad Mehrotra, and Naveen Ashish. 2007. Disambiguation algorithm for people search on the web. In *Proc. of IEEE International Conference on Data Engineering (IEEE ICDE)*.

Bradley Malin. 2005. Unsupervised name disambiguation via social network similarity. In *Workshop on Link Analysis, Counterterrorism, and Security*.

Gideon S. Mann and David Yarowsky. 2003. Unsupervised personal name disambiguation. In *Proceedings of the seventh conference on Natural Language Learning (CoNLL) at HLT-NAACL 2003*. ACL.

Gideon S. Mann. 2006. *Multi-Document Statistical Fact Extraction and Fusion*. Ph.D. thesis, Johns Hopkins University.

Hien T. Nguyen and Tru H. Cao, 2008. *Named Entity Disambiguation: A Hybrid Statistical and Rule-Based Incremental Approach*. Springer.

Octavian Popescu and Bernardo Magnini. 2007. Irst-bp: Web people search using name entities. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*. ACL.

Y. Ravin and Z. Kazi. 1999. Is hillary rodham clinton the president? disambiguating names across documents. In *Proceedings of the ACL '99 Workshop on Coreference and its Applications Association for Computational Linguistics*.

Horacio Saggion. 2008. Experiments on semantic-based clustering for cross-document coreference. In *International Joint Conference on Natural language Processing.*

Satoshi Sekine. 2008. Extended named entity ontology with attribute information. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*.

Amanda Spink, Bernard Jansen, and Jan Pedersen. 2004. Searching for people on web search engines. *Journal of Documentation*, 60:266 – 278.

Kazunari Sugiyama and Manabu Okumura. 2007. Titpi: Web people search task using semi-supervised clustering approach. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*. ACL.

Xiaojun Wan, Jianfeng Gao, Mu Li, and Binggong Ding. 2005. Person resolution in person search results: Webhawk. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM Press.

Investigation of Question Classifier in Question Answering

Zhiheng Huang
EECS Department
University of California
at Berkeley
CA 94720-1776, USA
zhiheng@cs.berkeley.edu

Marcus Thint
Intelligent Systems Research Center
British Telecom Group
Chief Technology Office
marcus.2.thint@bt.com

Asli Celikyilmaz
EECS Department
University of California
at Berkeley
CA 94720-1776, USA
asli@cs.berkeley.edu

Abstract

In this paper, we investigate how an accurate question classifier contributes to a question answering system. We first present a Maximum Entropy (ME) based question classifier which makes use of head word features and their WordNet hypernyms. We show that our question classifier can achieve the state of the art performance in the standard UIUC question dataset. We then investigate quantitatively the contribution of this question classifier to a feature driven question answering system. With our accurate question classifier and some standard question answer features, our question answering system performs close to the state of the art using TREC corpus.

1 Introduction

Question answering has drawn significant attention from the last decade (Prager, 2006). It attempts to answer the question posed in natural language by providing the answer phrase rather than the whole documents. An important step in question answering (QA) is to classify the question to the anticipated type of the answer. For example, the question of *Who discovered x-rays* should be classified into the type of human (individual). This information would narrow down the search space to identify the correct answer string. In addition, this information can suggest different strategies to search and verify a candidate answer. In fact, the combination of question classification and the named entity recognition is a key approach in modern question answering systems (Voorhees and Dang, 2005).

The question classification is by no means trivial: Simply using question wh-words can not achieve satisfactory results. The difficulty lies in classifying the *what* and *which* type questions. Considering the example *What is the capital of Yugoslavia*, it is of location (city) type, while *What is the pH scale* is of definition type. As with the previous work of (Li and Roth, 2002; Li and Roth, 2006; Krishnan et al., 2005; Moschitti et al., 2007), we propose a feature driven statistical question classifier (Huang et al., 2008). In particular, we propose head word feature and augment semantic features of such head words using WordNet. In addition, Lesk's word sense disambiguation (WSD) algorithm is adapted and the depth of hypernym feature is optimized. With further augment of other standard features such as unigrams, we can obtain accuracy of 89.0% using ME model for 50 fine classes over UIUC dataset.

In addition to building an accurate question classifier, we investigate the contribution of this question classifier to a feature driven question answering rank model. It is worth noting that, most of the features we used in question answering rank model, depend on the question type information. For instance, if a question is classified as a type of *sport*, we then only care about whether there are sport entities existing in the candidate sentences. It is expected that a fine grained named entity recognizer (NER) should make good use of the accurate question type information. However, due to the lack of a fine grained NER tool at hand, we employ the Stanford NER package (Finkel et al., 2005) which identifies only four types of named entities. Even with such a coarse named entity recognizer, the experiments show that the question classifier plays an important role in determining the performance of a question answering system.

The rest of the paper is organized as following. Section 2 reviews the maximum entropy model which are used in both question classification and question answering ranking. Section 3 presents the features used in question classification. Section 4 presents the question classification

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 543–550,
Singapore, 6-7 August 2009.

accuracy over UIUC question dataset. Section 5 presents the question answer features. Section 6 illustrates the results based on TREC question answer dataset. And Section 7 draws the conclusion.

2 Maximum Entropy Models

Maximum entropy (ME) models (Berger et al., 1996; Manning and Klein, 2003), also known as log-linear and exponential learning models, provide a general purpose machine learning technique for classification and prediction which has been successfully applied to natural language processing including part of speech tagging, named entity recognition etc. Maximum entropy models can integrate features from many heterogeneous information sources for classification. Each feature corresponds to a constraint on the model. Given a training set of (C, D), where C is a set of class labels and D is a set of feature represented data points, the maximal entropy model attempts to maximize the log likelihood

$$\log P(C|D,\lambda) = \sum_{(c,d)\in(C,D)} \log \frac{\exp \sum_i \lambda_i f_i(c,d)}{\sum_{c'} \exp \sum_j \lambda_j f_i(c,d)}, \quad (1)$$

where $f_i(c,d)$ are feature indicator functions. We use ME models for both question classification and question answer ranking. In question answer context, such function, for instance, could be the presence or absence of dictionary entities (as presented in Section 5.2) associated with a particular class type (either *true* or *false*, indicating a sentence can or cannot answer the question). λ_i are the parameters need to be estimated which reflects the importance of $f_i(c,d)$ in prediction.

3 Question Classification Features

Li and Roth (2002) have developed a machine learning approach which uses the SNoW learning architecture. They have compiled the UIUC question classification dataset[1] which consists of 5500 training and 500 test questions.[2] All questions in the dataset have been manually labeled according to the coarse and fine grained categories as shown in Table 1, with coarse classes (in bold) followed by their fine classes.

The UIUC dataset has laid a platform for the follow-up research including (Hacioglu and Ward, 2003; Zhang and Lee, 2003; Li and Roth, 2006; Krishnan et al., 2005; Moschitti et al., 2007). In contrast to Li and Roth (2006)'s approach which makes use of a very rich feature set, we propose to use a compact yet effective feature set. The features are briefly described as following. More detailed information can be found at (Huang et al., 2008).

Table 1: 6 coarse and 50 fine Question types defined in UIUC question dataset.

ABBR	letter	desc	**NUM**
abb	other	manner	code
exp	plant	reason	count
ENTITY	product	**HUMAN**	date
animal	religion	group	distance
body	sport	individual	money
color	substance	title	order
creative	symbol	desc	other
currency	technique	**LOC**	period
dis.med.	term	city	percent
event	vehicle	country	speed
food	word	mountain	temp
instrument	**DESC**	other	size
lang	definition	state	weight

Question wh-word The wh-word feature is the question wh-word in given questions. For example, the wh-word of question *What is the population of China* is *what*.

Head Word *head word* is defined as one *single* word specifying the object that the question seeks. For example the head word of *What is a group of turkeys called*, is *turkeys*. This is different to previous work including (Li and Roth, 2002; Krishnan et al., 2005) which has suggested a contiguous span of words (*a group of turkeys* in this example). The single word definition effectively avoids the noisy information brought by non-head word of the span (*group* in this case). A syntactic parser (Petrov and Klein, 2007) and the Collins rules (Collins, 1999) are modified to extract such head words.

WordNet Hypernym WordNet hypernyms are extracted for the head word of a given question. The classic Lesk algorithm (Lesk, 1986) is used to compute the most probable sense for a head word in the question context, and then the hypernyms are extracted based on that sense. The depth of hypernyms is set to

[1] Available at http://12r.cs.uiuc.edu/∼cogcomp/Data/QA/QC.
[2] Test questions are from TREC 10.

six with trial and error.[3] Hypernyms features capture the general terms of extracted head word. For instance, the head word of question *What is the proper name for a female walrus* is extracted as *walrus* and its direct hypernyms such as *mammal* and *animal* are extracted as informative features to predict the correct question type of ENTY:animal.

Unigram words Bag of words features. Such features provide useful question context information.

Word shape Five word shape features, namely all upper case, all lower case, mixed case, all digits, and other are used to serve as a coarse named entity recognizer.

4 Question Classification Experiments

We train a Maximum Entropy model using the UIUC 5500 training questions and test over the 500 test questions. Tables 2 shows the accuracy of 6 coarse class and 50 fine grained class, with features being fed incrementally. The question classification performance is measured by accuracy, i.e., the proportion of the correctly classified questions among all test questions. The baseline using the wh-head word results in 46.0% and 46.8% respectively for 6 coarse and 50 fine class classification. The incremental use of head word boosts the accuracy significantly to 92.2% and 82.0% for 6 and 50 classes. This reflects the informativeness of such feature. The inclusion of hypernym feature within 6 depths boosts 3.6% for 50 classes, while resulting in slight loss for 6 coarse classes. The further use of unigram feature leads to 2.8% gain in 50 classes. Finally, the use of word shape leads to 0.6% accuracy increase for 50 classes. The best accuracies achieved are 93.6% and 89.0% for 6 and 50 classes respectively.

Table 2: Question classification accuracy using incremental feature sets for 6 and 50 classes over UIUC split.

	6 class	50 class
wh-word	46.0	46.8
+ head word	92.2	82.0
+ hypernym	91.8	85.6
+ unigram	93.0	88.4
+ word shape	93.6	89.0

The individual feature contributions were discussed in greater detail in (Huang et al., 2008). Also, The SVM (rathern than ME model) was employed using the same feature set and the results were very close (93.4% for 6 class and 89.2% for 50 class). Table 3 shows the feature ablation experiment[4] which is missing in that paper. The experiment shows that the proposed head word and its hypernym features play an essential role in building an accurate question classifier.

Table 3: Question classification accuracy by removing one feature at a time for 6 and 50 classes over UIUC split.

	6 class	50 class
overall	93.6	89.0
- wh-word	93.6	89.0
- head word	92.8	88.2
- hypernym	90.8	84.2
- unigram	93.6	86.8
- word shape	93.0	88.4

Our best result feature space only consists of 13'697 binary features and each question has 10 to 30 active features. Compared to the over feature size of 200'000 in Li and Roth (2002), our feature space is much more compact, yet turned out to be more informative as suggested by the experiments. Table 4 shows the summary of the classification accuracy of all question classifiers which were applied to UIUC dataset.[5] Our results are summarized in the last row.

In addition, we have performed the 10 cross validation experiment over the 5500 UIUC training corpus using our best model. The result is 89.05 ± 1.25 and 83.73 ± 1.61 for 6 and 50 classes,[6] which outperforms the best result of 86.1 ± 1.1 for 6 classes as reported in (Moschitti et al., 2007).

5 Question Answer Features

For a pair of a question and a candidate sentence, we extract binary features which include CoNLL named entities presence feature (NE), dictionary

[3] We performed 10 cross validation experiment over training data and tried various depths of 1, 3, 6, 9 and ∞, with ∞ signifies that no depth constraint is imposed.

[4] Remove one feature at a time from the entire feature set.

[5] Note (1) that SNoW accuracy without the related word dictionary was not reported. With the semantically related word dictionary, it achieved 91%. Note (2) that SNoW with a semantically related word dictionary achieved 84.2% but the other algorithms did not use it.

[6] These results are worse than the result over UIUC split; as the UIUC test data includes a larger percentage of easily classified question types.

Table 4: Accuracy of all question classifiers which were applied to UIUC dataset.

Algorithm	6 class	50 class
Li and Roth, SNoW	$-^{(1)}$	$78.8^{(2)}$
Hacioglu et al., SVM+ECOC	–	80.2-82
Zhang & Lee, Linear SVM	87.4	79.2
Zhang & Lee, Tree SVM	90.0	–
Krishnan et al., SVM+CRF	93.4	86.2
Moschitti et al., Kernel	91.8	–
Maximum Entropy Model	93.6	89.0

entities presence feature (DIC), numerical entities presence feature (NUM), question specific feature (SPE), and dependency validity feature (DEP).

5.1 CoNLL named entities presence feature

We use Stanford named entity recognizer (NER) (Finkel et al., 2005) to identify CoNLL style NEs[7] as possible answer strings in a candidate sentence for a given type of question. In particular, if the question is ABBR type, we tag CoNLL LOC, ORG and MISC entities as candidate answers; If the question is HUMAN type, we tag CoNLL PER and ORG entities; And if the question is LOC type, we tag CoNLL LOC and MISC entities. For other types of questions, we assume there is no candidate CoNLL NEs to tag. We create a binary feature **NE** to indicate the presence or absence of tagged CoNLL entities. Further more, we create four binary features **NE-PER**, **NE-LOC**, **NE-ORG**, and **NE-MISC** to indicate the presence of tagged CoNLL PER, LOC, ORG and MISC entities.

5.2 Dictionary entities presence feature

As four types of CoNLL named entities are not enough to cover 50 question types, we include the 101 dictionary files compiled in the Ephyra project (Schlaefer et al., 2007). These dictionary files contain names for specific semantic types. For example, the *actor* dictionary comprises a list of actor names such as *Tom Hanks* and *Kevin Spacey*. For each question, if the head word of such question (see Section 3) matches the name of a dictionary file, then each noun phrase in a candidate sentence is looked up to check its presence in the dictionary. If so, a binary **DIC** feature is created. For example, for the question *What rank did Chester Nimitz reach*, as there is a *military rank* dictionary matches the head word *rank*, then all the noun phrases in a candidate sentence are looked up in the *military rank* dictionary. As a result, a sentence contains word *Admiral* will result in the DIC feature being activated, as such word is present in the *military rank* dictionary.

Note that an implementation tip is to allow the proximity match in the dictionary look up. Consider the question *What film introduced Jar Jar Binks*. As there is a match between the question head word *film* and the dictionary named *film*, each noun phrase in the candidate sentence is checked. However, no dictionary entities have been found from the candidate sentence *Best plays Jar Jar Binks, a floppy-eared, two-legged creature in "Star Wars: Episode I – The Phantom Menace"*, although there is movie entitled *Star Wars Episode I: The Phantom Menace* in the dictionary. Notice that *Star Wars: Episode I – The Phantom Menace* in the sentence and the dictionary entity *Star Wars Episode I: The Phantom Menace* do not have exactly identical spelling. The use of proximity look up which allows edit distance being less than 10% error can resolve this.

5.3 Numerical entities presence feature

There are so far no match for question types of NUM (as shown in Table 1) including NUM:count and NUM:date etc. These types of questions seek the numerical answers such as the amount of money and the duration of period. It is natural to compile regular expression patterns to match such entities. For example, for a NUM:money typed question *What is Rohm and Haas's annual revenue*, we compile NUM:money regular expression pattern which matches the strings of number followed by a currency sign (*$* and *dollars* etc). Such pattern is able to identify *4 billion $* as a candidate answer in the candidate sentence *Rohm and Haas, with 4 billion $ in annual sales...* There are 13 patterns compiled to cover all numerical types. We create a binary feature **NUM** to indicate the presence of possible numerical answers in a sentence.

5.4 Specific features

Specific features are question dependent. For example, for question *When was James Dean born*, any candidate sentence matches the pattern *James Dean (number - number)* is likely to answer such question. We create a binary feature **SPE** to indicate the presence of such match between a ques-

[7] Person (PER), location (LOC), organization (ORG), and miscellaneous (MISC).

tion and a candidate sentence. We list all question and sentence match patterns which are used in our experiments as following:

when born feature 1 The question begins with *when is/was* and follows by a person name and then follows by key word *born*; The candidate sentence contains such person name which follows by the pattern of *(number - number)*.

when born feature 2 The question begins with *when is/was* and follows by a person name and then follows by key word *born*; The candidate sentence contains such person name, a NUM:date entity, and a key word *born*.

where born feature 1 The question begins with *where is/was* and follows by a person name and then follows by key word *born*; The candidate sentence contains such person name, a NER LOC entity, and a key word *born*.

when die feature 1 The question begins with *when did* and follows by a person name and then follows by key word *die*; The candidate sentence contains such person name which follows by the pattern of *(number - number)*.

when die feature 2 The question begins with *when did* and follows by a person name and then follows by key word *die*; The candidate sentence contains such person name, a NUM:date entity, and a key word *died*.

how many feature The question begins with *how many* and follows by a noun; The candidate sentence contains a number and then follows by such noun.

cooccurrent Feature This feature takes two phrase arguments, if the question contains the first phrase and the candidate sentence contains the second, such feature would be activated.

Note that the construction of specific features require the access to aforementioned extracted named entities. For example, the **when born feature 2** pattern needs the information whether a candidate sentence contains a NUM:date entity and **where born feature 1** pattern needs the information whether a candidate sentence contains a NER LOC entity. Note also that the patterns of **when born feature** and **when die feature** have similar structure and thus can be simplified in implementation. **How many feature** can be used to identify the sentence *Amtrak annually serves about 21 million passengers* for question *How many passengers does Amtrak serve annually*. The **cooccurrent feature** is the most general one. An example of cooccurrent feature would take the arguments of *marry* and *husband*, or *marry* and *wife*. Such feature would be activated for question *Whom did Eileen Marie Collins marry* and candidate sentence *... were Collins' husband, Pat Youngs, an airline pilot...* It is worth noting that the two arguments are not necessarily different. For example, they could be both *established*, which makes such feature activated for question *When was the IFC established* and candidate sentence *IFC was established in 1956 as a member of the World Bank Group*. The reason why we use the cooccurrence of the word *established* is due to its main verb role, which may carry more information than other words.

5.5 Dependency validity features

Like (Cui et al., 2004), we extract the dependency path from the question word to the common word (existing in both question and sentence), and the path from candidate answer (such as CoNLL NE and numerical entity) to the common word for each pair of question and candidate sentence using Stanford dependency parser (Klein and Manning, 2003; Marneffe et al., 2006). For example, for question *When did James Dean die* and candidate sentence *In 1955, actor James Dean was killed in a two-car collision near Cholame, Calif.*, we extract the pathes of *When:advmod:nsubj:Dean* and *1955:prep-in:nsubjpass:Dean* for question and sentence respectively, where *advmod* and *nsubj* etc. are grammatical relations. We propose the dependency validity feature (**DEP**) as following. For all paired paths between a question and a candidate sentence, if at least one pair of path in which all pairs of grammatical relations have been seen in the training, then the DEP feature is set to be true, false otherwise. That is, the true validity feature indicates that at least one pair of path between the question and candidate sentence is possible to be a true pair (ie, the candidate noun phrase in the sentence path is the true answer).

6 Question Answer Experiments

Recall that most of the question answer features depend on the question classifier. For instance, the NE feature checks the presence or absence of CoNLL style named entities subject to the classified question type. In this section, we evaluate how the quality of question classifiers affects the question answering performance.

6.1 Experiment setup

We use TREC99-03 factoid questions for training and TREC04 factoid questions for testing. To facilitate the comparison to others work (Cui et al., 2004; Shen and Klakow, 2006), we first retrieve all relevant documents which are compiled by Ken Litkowski[8] to create training and test datasets. We

[8] Available at http://trec.nist.gov/data/qa.html.

then apply key word search for each question and retrieve the top 20 relevant sentences. We create a feature represented data point using each pair of question and candidate sentence and label it either *true* or *false* depending on whether the sentence can answer the given question or not. The labeling is conducted by matching the gold factoid answer pattern against the candidate sentence.

There are two extra steps performed for training set but not for test data. In order to construct a high quality training set, we manually check the correctness of the training data points and remove the false positive ones which cannot support the question although there is a match to gold answer. In addition, in order to keep the training data well balanced, we keep maximum four false data points (question answer pair) for each question but no limit over the true label data points. In doing so, we use 1458 questions to compile 8712 training data points and among them 1752 have true labels. Similarly, we use 202 questions to compile 4008 test data points and among them 617 have true labels.

We use the training data to train a maximum entropy model and use such model to rank test data set. Compared with a classification task (such as the question classifier), the ranking process requires one extra step: For data points which share the same question, the probabilities of being predicted as true label are used to rank the data points. In align with the previous work, performance is evaluated using mean reciprocal rank (MRR), top 1 prediction accuracy (top1) and top 5 prediction accuracy (top5). For the test data set, 157 among the 202 questions have correct answers found in retrieved sentences. This leads to the upper bound of MRR score being 77.8%.

To evaluate how the quality of question classifiers affects the question answering, we have created three question classifiers: QC1, QC2 and QC3. The features which are used to train these question classifiers and their performance are shown in Table 5. Note that QC3 is the best question classifier we obtained in Section 4.

Table 5: Features used to train and the performance of three question classifiers.

Name	features	6 class	50 class
QC1	wh-word	46.0	46.8
QC2	wh-word+ head	92.2	82.0
QC3	All	93.6	89.0

6.2 Experiment results

The first experiment is to evaluate the individual contribution of various features derived using three question classifiers. Table 6 shows the baseline result and results using DIC, NE, NE-4, REG, SPE, and DEP features. The baseline is the key word search without the use of maximum entropy model. As can be seen, the question classifiers do not affect the DIC feature at all, as DIC feature does not depend on question classifiers. Better question classifier boosts considerable gain for NE, NE-4 and REG in their contribution to question answering. For example, the best question classifier QC3 outperforms the worst one (QC1) by 1.5%, 2.0%, and 2.0% MRR scores for NE, NE-4 and REG respectively. However, it is surprising that the MRR and top5 contribution of NE and NE-4 decreases if QC1 is replaced by QC2, although the top1 score results in performance gain slightly. This unexpected results can be partially explained as follows. For some questions, even QC2 produces correct predictions, the errors of NE and NE-4 features may cause over-confident scores for certain candidate sentences. As SPE and DEP are not directly dependent on question classifier, their individual contribution only changes slightly or remains the same for different question classifiers. If the best question classifier is used, the most important features are SPE and REG, which can individually boost the MRR score over 54%, while the others result in less significant gains.

We now incrementally use various features and the results are show in Table 6 as well. As can be seen, the more features and the better question classifier are used, the higher performance the ME model has. The inclusion of REG and SPE results in significant boost for the performance. For example, if the best question classifier QC3 is used, the REG results in 6.9% and 8% gain for MRR and top1 scores respectively. This is due to a large portion of NUM type questions in test dataset. The SPE feature contributes significantly to the performance due to its high precision in answering birth/death time/location questions. NE and NE-4 result in reasonable gains while DEP feature contributes little. However, this does not mean that DEP is not important, as once the model reaches a high MRR score, it becomes hard to improve.

Table 6 clearly shows that the question type classifier plays an essential role in a high perfor-

Table 6: Performance of individual and incremental feature sets for three question classifiers.

Feature	MRR			Top1			Top5		
	QC1	QC2	QC3	QC1	QC2	QC3	QC1	QC2	QC3
Individual									
Baseline	49.9	49.9	49.9	40.1	40.1	40.1	59.4	59.4	59.4
DIC	49.5	49.5	49.5	42.6	42.6	42.6	60.4	60.4	60.4
NE	48.5	47.5	50.0	40.6	40.6	42.6	61.9	60.9	63.4
NE-4	49.5	48.5	51.5	41.6	42.1	44.6	62.4	61.9	64.4
REG	52.0	54.0	54.0	44.1	47.0	47.5	64.4	65.3	65.3
SPE	55.0	55.0	55.0	48.5	48.5	48.5	64.4	64.4	64.4
DEP	51.0	51.5	52.0	43.6	44.1	44.6	65.3	65.8	65.8
Incremental									
Baseline	49.9	49.9	49.9	40.1	40.1	40.1	59.4	59.4	59.4
+DIC	49.5	49.5	49.5	42.6	42.6	42.6	60.4	60.4	60.4
+NE	50.0	48.5	51.0	43.1	42.1	44.6	62.9	61.4	64.4
+NE-4	51.5	50.0	53.0	44.1	43.6	46.0	63.4	62.9	65.8
+REG	55.0	56.9	59.9	48.0	51.0	54.0	68.3	68.8	71.8
+SPE	60.4	62.4	65.3	55.4	58.4	61.4	70.8	70.8	73.8
+DEP	61.4	62.9	**66.3**	55.9	58.4	**62.4**	71.8	71.8	**73.8**

mance question answer system. Assume all the features are used, the better question classifier significantly boosts the overall performance. For example, the best question classifier QC3 outperforms the worst QC1 by 4.9%, 6.5%, and 2.0% for MRR, top1 and top5 scores respectively. Even compared to a good question classifier QC2, the gain of using QC3 is still 3.4%, 4.0% and 2.0% for MRR, top1 and top5 scores respectively. One can imagine that if a fine grained NER is available (rather than the current four type coarse NER), the potential gain is much significant.

The reason that the question classifier affects the question answering performance is straightforward. As a upstream source, the incorrect classification of question type would confuse the downstream answer search process. For example, for question *What is Rohm and Haas's annual revenue*, our best question classifier is able to classify it into the correct type of NUM:money and thus would put *$ 4 billion* as a candidate answer. However, the inferior question classifiers misclassify it into HUM:ind type and thereby could not return a correct answer. Figure 1 shows the individual MRR scores for the 42 questions (among the 202 test questions) which have different predicted question types using QC3 and QC2. For almost all test questions, the accurate question classifier QC3 achieves higher MRR scores compared to QC2.

Table 7 shows performance of various question answer systems including (Tanev et al., 2004; Wu et al., 2005; Cui et al., 2004; Shen and Klakow, 2006) and this work which were applied to the same training and test datasets. Among all the systems, our model can achieve the best MRR score of 66.3%, which is close to the state of the art of 67.0%. Considering the question answer features used in this paper are quite standard, the boost is mainly due to our accurate question classifier.

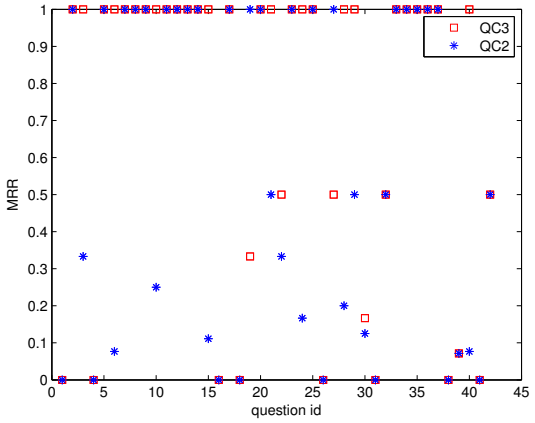


Figure 1: Individual MRR scores for questions which have different predicted question types using QC3 and QC2.

Table 7: Various system performance comparison.

System	MRR	Top1	Top5
Tanev et al. 2004	57.0	49.0	67.0
Cui et al. 2004	60.0	53.0	70.0
Shen and Klakow, 2006	67.0	62.0	74.0
This work	66.3	62.4	73.8

7 Conclusion

In this paper, we have presented a question classifier which makes use of a compact yet efficient feature set. The question classifier outperforms previous question classifiers over the standard UIUC question dataset. We further investigated quantitatively how the quality of question classifier impacts the performance of question answer system. The experiments showed that an accurate question classifier plays an essential role in question answering system. With our accurate question classifier and some standard question answer features, our question answering system performs close to the state of the art.

Acknowledgments

We wish to thank the three anonymous reviewers for their invaluable comments. This research was supported by British Telecom grant CT1080028046 and BISC Program of UC Berkeley.

References

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

M. Collins. 1999. Head-driven statistical models for natural language parsing. *PhD thesis*, University of Pennsylvania.

H. Cui, K Li, R. Sun, T. Chua, and M. Kan. 2004. National university of singapore at the trec-13 question answering. In *Proc. of TREC 2004, NIST*.

J. Finkel, T. Grenager, and C. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proc. of ACL*, pages 363-370.

D. Hacioglu and W. Ward. 2003. Question classification with support vector machines and error correcting codes. In *Proc. of the ACL/HLT*, vol. 2, pages 28–30.

Z. Huang, M. Thint, and Z. Qin. 2008. Question classification using head words and their hypernyms. In *Proc. of the EMNLP*.

D. Klein and C. D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL 2003*, vol. 1, pages 423–430.

V. Krishnan, S. Das, and S. Chakrabarti. 2005. Enhanced answer type inference from questions using sequential models. In *Proc. of the HLT/EMNLP*.

M. Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *ACM Special Interest Group for Design of Communication Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26.

X. Li and D. Roth. 2002. Learning question classifiers. In *the 19th international conference on Computational linguistics*, vol. 1, pages 1-7.

X. Li and D. Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249.

C. D. Manning and D. Klein. 2003. Optimization, maxent models, and conditional estimation without magic. *Tutorial at HLT-NAACL 2003 and ACL 2003*.

M. D. Marneffe, B. MacCartney and C. D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. of LREC 2006*.

A. Moschitti, S. Quarteroni, R. Basili and S. Manandhar 2007. Exploiting syntactic and shallow semantic kernels for question answer classification. In *Proc. of ACL 2007*, pages 776-783.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. of the HLT-NAACL.*

J. Prager. 2006. Open-domain question-answering. In *Foundations and Trends in Information Retrieval*, vol. 1, pages 91-231, 2006.

N. Schlaefer, J. Ko, J. Betteridge, G. Sautter, M. Pathak and E. Nyberg. 2007. Semantic extensions of the Ephyra QA system for TREC 2007. In *Proc. of the TREC 2007*.

D. Shen and D. Klakow. 2006. Exploring correlation of dependency relation paths for answer extraction. In *Proc. of the ACL 2006*.

H. Tanev, M. Kouylekov, and B. Magnini. 2004. Combining linguistic processing and web mining for question answering: Itc-irst at TREC-2004. In *Proc. of the TREC 2004, NIST*.

E. M. Voorhees and H. T. Dang. 2005. Overview of the TREC 2005 question answering track. In *Proc. of the TREC 2005, NIST*.

M. Wu, M. Duan, S. Shaikh, S. Small, and T. Strzalkowski. 2005. University at Albanys ILQUA in TREC 2005. In *Proc. of the TREC 2005, NIST*.

D. Zhang and W. S. Lee. 2003. Question classification using support vector machines. In *The ACM SIGIR conference in information retrieval*, pages 26–32.

An Empirical Study of Semi-supervised Structured Conditional Models for Dependency Parsing

Jun Suzuki, Hideki Isozaki
NTT CS Lab., NTT Corp.
Kyoto, 619-0237, Japan
jun@cslab.kecl.ntt.co.jp
isozaki@cslab.kecl.ntt.co.jp

Xavier Carreras, and Michael Collins
MIT CSAIL
Cambridge, MA 02139, USA
carreras@csail.mit.edu
mcollins@csail.mit.edu

Abstract

This paper describes an empirical study of high-performance dependency parsers based on a semi-supervised learning approach. We describe an extension of semi-supervised structured conditional models (SS-SCMs) to the dependency parsing problem, whose framework is originally proposed in (Suzuki and Isozaki, 2008). Moreover, we introduce two extensions related to dependency parsing: The first extension is to combine SS-SCMs with another semi-supervised approach, described in (Koo et al., 2008). The second extension is to apply the approach to second-order parsing models, such as those described in (Carreras, 2007), using a two-stage semi-supervised learning approach. We demonstrate the effectiveness of our proposed methods on dependency parsing experiments using two widely used test collections: the Penn Treebank for English, and the Prague Dependency Treebank for Czech. Our best results on test data in the above datasets achieve 93.79% parent-prediction accuracy for English, and 88.05% for Czech.

1 Introduction

Recent work has successfully developed dependency parsing models for many languages using supervised learning algorithms (Buchholz and Marsi, 2006; Nivre et al., 2007). Semi-supervised learning methods, which make use of unlabeled data in addition to labeled examples, have the potential to give improved performance over purely supervised methods for dependency parsing. It is often straightforward to obtain large amounts of unlabeled data, making semi-supervised approaches appealing; previous work on semi-supervised methods for dependency parsing includes (Smith and Eisner, 2007; Koo et al., 2008; Wang et al., 2008).

In particular, Koo et al. (2008) describe a semi-supervised approach that makes use of cluster features induced from unlabeled data, and gives state-of-the-art results on the widely used dependency parsing test collections: the Penn Treebank (PTB) for English and the Prague Dependency Treebank (PDT) for Czech. This is a very simple approach, but provided significant performance improvements comparing with the state-of-the-art supervised dependency parsers such as (McDonald and Pereira, 2006).

This paper introduces an alternative method for semi-supervised learning for dependency parsing. Our approach basically follows a framework proposed in (Suzuki and Isozaki, 2008). We extend it for dependency parsing, which we will refer to as a Semi-supervised Structured Conditional Model (SS-SCM). In this framework, a structured conditional model is constructed by incorporating a series of generative models, whose parameters are estimated from unlabeled data. This paper describes a basic method for learning within this approach, and in addition describes two extensions. The first extension is to combine our method with the cluster-based semi-supervised method of (Koo et al., 2008). The second extension is to apply the approach to second-order parsing models, more specifically the model of (Carreras, 2007), using a two-stage semi-supervised learning approach.

We conduct experiments on dependency parsing of English (on Penn Treebank data) and Czech (on the Prague Dependency Treebank). Our experiments investigate the effectiveness of: 1) the basic SS-SCM for dependency parsing; 2) a combination of the SS-SCM with Koo et al. (2008)'s semi-supervised approach (even in the case we used the same unlabeled data for both methods); 3) the two-stage semi-supervised learning approach that in-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 551–560,
Singapore, 6-7 August 2009.

corporates a second-order parsing model. In addition, we evaluate the SS-SCM for English dependency parsing with large amounts (up to 3.72 billion tokens) of unlabeled data .

2 Semi-supervised Structured Conditional Models for Dependency Parsing

Suzuki et al. (2008) describe a semi-supervised learning method for conditional random fields (CRFs) (Lafferty et al., 2001). In this paper we extend this method to the dependency parsing problem. We will refer to this extended method as *Semi-supervised Structured Conditional Models (SS-SCMs)*. The remainder of this section describes our approach.

2.1 The Basic Model

Throughout this paper we will use $\mathbf{x}$ to denote an input sentence, and $\mathbf{y}$ to denote a labeled dependency structure. Given a sentence $\mathbf{x}$ with n words, a labeled dependency structure $\mathbf{y}$ is a set of n dependencies of the form (h, m, l), where h is the index of the head-word in the dependency, m is the index of the modifier word, and l is the label of the dependency. We use $h = 0$ for the root of the sentence. We assume access to a set of labeled training examples, $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, and in addition a set of unlabeled examples, $\{\mathbf{x}'_i\}_{i=1}^M$.

In conditional log-linear models for dependency parsing (which are closely related to conditional random fields (Lafferty et al., 2001)), a distribution over dependency structures for a sentence $\mathbf{x}$ is defined as follows:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\{g(\mathbf{x}, \mathbf{y})\}, \qquad (1)$$

where $Z(\mathbf{x})$ is the partition function, $\mathbf{w}$ is a parameter vector, and

$$g(\mathbf{x}, \mathbf{y}) = \sum_{(h,m,l) \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m, l)$$

Here $\mathbf{f}(\mathbf{x}, h, m, l)$ is a feature vector representing the dependency (h, m, l) in the context of the sentence $\mathbf{x}$ (see for example (McDonald et al., 2005a)).

In this paper we extend the definition of $g(\mathbf{x}, \mathbf{y})$ to include features that are induced from unlabeled data. Specifically, we define

$$g(\mathbf{x}, \mathbf{y}) = \sum_{(h,m,l) \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m, l) + \sum_{(h,m,l) \in \mathbf{y}} \sum_{j=1}^{k} v_j q_j(\mathbf{x}, h, m, l). \quad (2)$$

In this model $v_1, \ldots, v_k$ are scalar parameters that may be positive or negative; $q_1 \ldots q_k$ are functions (in fact, generative models), that are trained on unlabeled data. The v_j parameters will dictate the relative strengths of the functions $q_1 \ldots q_k$, and will be trained on labeled data.

For convenience, we will use $\mathbf{v}$ to refer to the vector of parameters $v_1 \ldots v_k$, and $\mathbf{q}$ to refer to the set of generative models $q_1 \ldots q_k$. The full model is specified by values for $\mathbf{w}, \mathbf{v}$, and $\mathbf{q}$. We will write $p(\mathbf{y}|\mathbf{x}; \mathbf{w}, \mathbf{v}, \mathbf{q})$ to refer to the conditional distribution under parameter values $\mathbf{w}, \mathbf{v}, \mathbf{q}$.

We will describe a three-step parameter estimation method that: 1) initializes the $\mathbf{q}$ functions (generative models) to be uniform distributions, and estimates parameter values $\mathbf{w}$ and $\mathbf{v}$ from labeled data; 2) induces new functions $q_1 \ldots q_k$ from unlabeled data, based on the distribution defined by the $\mathbf{w}, \mathbf{v}, \mathbf{q}$ values from step (1); 3) re-estimates $\mathbf{w}$ and $\mathbf{v}$ on the labeled examples, keeping the $q_1 \ldots q_k$ from step (2) fixed. The end result is a model that combines supervised training with generative models induced from unlabeled data.

2.2 The Generative Models

We now describe how the generative models $q_1 \ldots q_k$ are defined, and how they are induced from unlabeled data. These models make direct use of the feature-vector definition $\mathbf{f}(\mathbf{x}, \mathbf{y})$ used in the original, fully supervised, dependency parser.

The first step is to partition the d features in $\mathbf{f}(\mathbf{x}, \mathbf{y})$ into k separate feature vectors, $\mathbf{r}_1(\mathbf{x}, \mathbf{y}) \ldots \mathbf{r}_k(\mathbf{x}, \mathbf{y})$ (with the result that $\mathbf{f}$ is the concatenation of the k feature vectors $\mathbf{r}_1 \ldots \mathbf{r}_k$). In our experiments on dependency parsing, we partitioned $\mathbf{f}$ into up to over 140 separate feature vectors corresponding to different feature types. For example, one feature vector $\mathbf{r}_j$ might include only those features corresponding to word bigrams involved in dependencies (i.e., indicator functions tied to the word bigram (x_m, x_h) involved in a dependency $(\mathbf{x}, h, m, l)$).

We then define a generative model that assigns a probability

$$q'_j(\mathbf{x}, h, m, l) = \prod_{a=1}^{d_j} \theta_{j,a}^{r_{j,a}(\mathbf{x},h,m,l)} \qquad (3)$$

to the d_j-dimensional feature vector $\mathbf{r}_j(\mathbf{x}, h, m, l)$. The parameters of this model are $\theta_{j,1} \ldots \theta_{j,d_j}$;

they form a multinomial distribution, with the constraints that $\theta_{j,a} \geq 0$, and $\sum_a \theta_{j,a} = 1$. This model can be viewed as a very simple (naive-Bayes) model that defines a distribution over feature vectors $\mathbf{r}_j \in \mathbb{R}^{d_j}$. The next section describes how the parameters $\theta_{j,a}$ are trained on unlabeled data.

Given parameters $\theta_{j,a}$, we can simply define the functions $q_1 \ldots q_k$ to be log probabilities under the generative model:

$$\begin{aligned} q_j(\mathbf{x}, h, m, l) &= \log q'_j(\mathbf{x}, h, m, l) \\ &= \sum_{a=1}^{d_j} r_{j,a}(\mathbf{x}, h, m, l) \log \theta_{j,a}. \end{aligned}$$

We modify this definition slightly, be introducing scaling factors $c_{j,a} > 0$, and defining

$$q_j(\mathbf{x}, h, m, l) = \sum_{a=1}^{d_j} r_{j,a}(\mathbf{x}, h, m, l) \log \frac{\theta_{j,a}}{c_{j,a}} \quad (4)$$

In our experiments, $c_{j,a}$ is simply a count of the number of times the feature indexed by (j, a) appears in unlabeled data. Thus more frequent features have their contribution down-weighted in the model. We have found this modification to be beneficial.

2.3 Estimating the Parameters of the Generative Models

We now describe the method for estimating the parameters $\theta_{j,a}$ of the generative models. We assume initial parameters $\mathbf{w}, \mathbf{v}, \mathbf{q}$, which define a distribution $p(\mathbf{y}|\mathbf{x}'_i; \mathbf{w}, \mathbf{v}, \mathbf{q})$ over dependency structures for each unlabeled example $\mathbf{x}'_i$. We will re-estimate the generative models $\mathbf{q}$, based on unlabeled examples. The likelihood function on unlabeled data is defined as

$$\sum_{i=1}^{M} \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}'_i; \mathbf{w}, \mathbf{v}, \mathbf{q}) \sum_{(h,m,l) \in \mathbf{y}} \log q'_j(\mathbf{x}'_i, h, m, l), \quad (5)$$

where q'_j is as defined in Eq. 3. This function resembles the Q function used in the EM algorithm, where the hidden labels (in our case, dependency structures), are filled in using the conditional distribution $p(\mathbf{y}|\mathbf{x}'_i; \mathbf{w}, \mathbf{v}, \mathbf{q})$.

It is simple to show that the estimates $\theta_{j,a}$ that maximize the function in Eq. 5 can be defined as follows. First, define a vector of *expected counts* based on $\mathbf{w}, \mathbf{v}, \mathbf{q}$ as

$$\hat{\mathbf{r}}_j = \sum_{i=1}^{M} \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}'_i; \mathbf{w}, \mathbf{v}, \mathbf{q}) \sum_{(h,m,l) \in \mathbf{y}} \mathbf{r}_j(\mathbf{x}'_i, h, m, l).$$

Note that it is straightforward to calculate these expected counts using a variant of the inside-outside algorithm (Baker, 1979) applied to the (Eisner, 1996) dependency-parsing data structures (Paskin, 2001) for projective dependency structures, or the matrix-tree theorem (Koo et al., 2007; Smith and Smith, 2007; McDonald and Satta, 2007) for non-projective dependency structures.

The estimates that maximize Eq. 5 are then

$$\theta_{j,a} = \frac{\hat{r}_{j,a}}{\sum_{a=1}^{d_j} \hat{r}_{j,a}}.$$

In a slight modification, we employ the following estimates in our model, where $\eta > 1$ is a parameter of the model:

$$\theta_{j,a} = \frac{(\eta - 1) + \hat{r}_{j,a}}{d_j \times (\eta - 1) + \sum_{a=1}^{d_j} \hat{r}_{j,a}}. \quad (6)$$

This corresponds to a MAP estimate under a Dirichlet prior over the $\theta_{j,a}$ parameters.

2.4 The Complete Parameter-Estimation Method

This section describes the full parameter estimation method. The input to the algorithm is a set of labeled examples $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, a set of unlabeled examples $\{\mathbf{x}'_i\}_{i=1}^M$, a feature-vector definition $\mathbf{f}(\mathbf{x}, \mathbf{y})$, and a partition of $\mathbf{f}$ into k feature vectors $\mathbf{r}_1 \ldots \mathbf{r}_k$ which underly the generative models. The output from the algorithm is a parameter vector $\mathbf{w}$, a set of generative models $q_1 \ldots q_k$, and parameters $v_1 \ldots v_k$, which define a probabilistic dependency parsing model through Eqs. 1 and 2. The learning algorithm proceeds in three steps:

Step 1: Estimation of a Fully Supervised Model. We choose the initial value $\mathbf{q}^0$ of the generative models to be the uniform distribution, i.e., we set $\theta_{j,a} = 1/d_j$ for all j, a. We then define the regularized log-likelihood function for the labeled examples, with the generative model fixed at $\mathbf{q}^0$, to be:

$$\begin{aligned} L(\mathbf{w}, \mathbf{v}; \mathbf{q}^0) = &\sum_{i=1}^{n} \log p(\mathbf{y}_i|\mathbf{x}_i; \mathbf{w}, \mathbf{v}, \mathbf{q}^0) \\ &- \frac{C}{2} \left(||\mathbf{w}||^2 + ||\mathbf{v}||^2 \right) \end{aligned}$$

This is a conventional regularized log-likelihood function, as commonly used in CRF models. The parameter $C > 0$ dictates the level of regularization in the model. We define the initial parameters $(\mathbf{w}^0, \mathbf{v}^0) = \arg\max_{\mathbf{w},\mathbf{v}} L(\mathbf{w}, \mathbf{v}; \mathbf{q}^0)$. These parameters can be found using conventional methods for estimating the parameters of regularized log-likelihood functions (in our case we use LBFGS (Liu and Nocedal, 1989)). Note that the gradient of the log-likelihood function can be calculated using the inside-outside algorithm applied to projective dependency parse structures, or the matrix-tree theorem applied to non-projective structures.

Step 2: Estimation of the Generative Models. In this step, expected count vectors $\hat{\mathbf{r}}_1 \ldots \hat{\mathbf{r}}_k$ are first calculated, based on the distribution $p(\mathbf{y}|\mathbf{x}; \mathbf{w}^0, \mathbf{v}^0, \mathbf{q}^0)$. Generative model parameters $\theta_{j,a}$ are calculated through the definition in Eq. 6; these estimates define updated generative models q_j^1 for $j = 1 \ldots k$ through Eq. 4. We refer to the new values for the generative models as $\mathbf{q}^1$.

Step 3: Re-estimation of $\mathbf{w}$ and $\mathbf{v}$. In the final step, $\mathbf{w}^1$ and $\mathbf{v}^1$ are estimated as $\arg\max_{\mathbf{w},\mathbf{v}} L(\mathbf{w}, \mathbf{v}; \mathbf{q}^1)$ where $L(\mathbf{w}, \mathbf{v}; \mathbf{q}^1)$ is defined in an analogous way to $L(\mathbf{w}, \mathbf{v}; \mathbf{q}^0)$. Thus $\mathbf{w}$ and $\mathbf{v}$ are re-estimated to optimize log-likelihood of the labeled examples, with the generative models $\mathbf{q}^1$ estimated in step 2.

The final output from the algorithm is the set of parameters $(\mathbf{w}^1, \mathbf{v}^1, \mathbf{q}^1)$. Note that it is possible to iterate the method—steps 2 and 3 can be repeated multiple times (Suzuki and Isozaki, 2008)—but in our experiments we only performed these steps once.

3 Extensions

3.1 Incorporating Cluster-Based Features

Koo et al. (2008) describe a semi-supervised approach that incorporates cluster-based features, and that gives competitive results on dependency parsing benchmarks. The method is a two-stage approach. First, hierarchical word clusters are derived from unlabeled data using the Brown et al. clustering algorithm (Brown et al., 1992). Second, a new feature set is constructed by representing words by bit-strings of various lengths, corresponding to clusters at different levels of the hierarchy. These features are combined with conventional features based on words and part-of-speech tags. The new feature set is then used within a conventional discriminative, supervised approach, such as the averaged perceptron algorithm.

The important point is that their approach uses unlabeled data only for the construction of a new feature set, and never affects to learning algorithms. It is straightforward to incorporate cluster-based features within the SS-SCM approach described in this paper. We simply use the cluster-based feature-vector representation $\mathbf{f}(\mathbf{x}, \mathbf{y})$ introduced by (Koo et al., 2008) as the basis of our approach.

3.2 Second-order Parsing Models

Previous work (McDonald and Pereira, 2006; Carreras, 2007) has shown that second-order parsing models, which include information from "sibling" or "grandparent" relationships between dependencies, can give significant improvements in accuracy over first-order parsing models. In principle it would be straightforward to extend the SS-SCM approach that we have described to second-order parsing models. In practice, however, a bottleneck for the method would be the estimation of the generative models on unlabeled data. This step requires calculation of marginals on unlabeled data. Second-order parsing models generally require more costly inference methods for the calculation of marginals, and this increased cost may be prohibitive when large quantities of unlabeled data are employed.

We instead make use of a simple 'two-stage' approach for extending the SS-SCM approach to the second-order parsing model of (Carreras, 2007). In the first stage, we use a first-order parsing model to estimate generative models $q_1 \ldots q_k$ from unlabeled data. In the second stage, we incorporate these generative models as features within a second-order parsing model. More precisely, in our approach, we first train a first-order parsing model by Step 1 and 2, exactly as described in Section 2.4, to estimate $\mathbf{w}^0$, $\mathbf{v}^0$ and $\mathbf{q}^1$. Then, we substitute Step 3 as a supervised learning such as MIRA with a second-order parsing model (McDonald et al., 2005a), which incorporates $\mathbf{q}^1$ as a real-values features. We refer this two-stage approach to as **two-stage SS-SCM**.

In our experiments we use the 1-best MIRA algorithm (McDonald and Pereira, 2006)[1] as a

[1]We used a slightly modified version of 1-best MIRA, whose difference can be found in the third line in Eq. 7, namely, including $L(\mathbf{y}_i, \mathbf{y})$.

(a) English dependency parsing

Data set	(WSJ Sec. IDs)	# of sentences	# of tokens
Training	(02–21)	39,832	950,028
Development	(22)	1,700	40,117
Test	(23)	2,012	47,377
Unlabeled		1,796,379	43,380,315

(b) Czech dependency parsing

Data set	# of sentences	# of tokens
Training	73,088	1,255,590
Development	7,507	126,030
Test	7,319	125,713
Unlabeled	2,349,224	39,336,570

Table 1: Details of training, development, test data (labeled data sets) and unlabeled data used in our experiments

Corpus	article name (mm/yy)		# of sent.	# of tokens
BLLIP	wsj	00/87–00/89	1,796,379	43,380,315
Tipster	wsj	04/90–03/92	1,550,026	36,583,547
North	wsj	07/94–12/96	2,748,803	62,937,557
American	reu	04/94–07/96	4,773,701	110,001,109
Reuters	reu	09/96–08/97	12,969,056	214,708,766
English	afp	05/94–12/06	21,231,470	513,139,928
Gigaword	apw	11/94–12/06	46,978,725	960,733,303
	ltw	04/94–12/06	10,524,545	230,370,454
	nyt	07/94–12/06	60,752,363	1,266,531,274
	xin	01/95–12/06	12,624,835	283,579,330
total			175,949,903	3,721,965,583

Table 2: Details of the larger unlabeled data set used in English dependency parsing: sentences exceeding 128 tokens in length were excluded for computational reasons.

parameter-estimation method for the second-order parsing model. In particular, we perform the following optimizations on each update $t = 1, ..., T$ for re-estimating $\mathbf{w}$ and $\mathbf{v}$:

$$\begin{aligned} &\min ||\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}|| + ||\mathbf{v}^{(t+1)} - \mathbf{v}^{(t)}|| \\ &\text{s.t.} \quad S(\mathbf{x}_i, \mathbf{y}_i) - S(\mathbf{x}_i, \hat{\mathbf{y}}) \geq L(\mathbf{y}_i, \hat{\mathbf{y}}) \\ &\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} S(\mathbf{x}_i, \mathbf{y}) + L(\mathbf{y}_i, \mathbf{y}), \end{aligned} \tag{7}$$

where $L(\mathbf{y}_i, \mathbf{y})$ represents the loss between correct output of i'th sample $\mathbf{y}_i$ and $\mathbf{y}$. Then, the scoring function S for each $\mathbf{y}$ can be defined as follows:

$$\begin{aligned} S(\mathbf{x}, \mathbf{y}) = &\mathbf{w} \cdot (\mathbf{f}_1(\mathbf{x}, \mathbf{y}) + \mathbf{f}_2(\mathbf{x}, \mathbf{y})) \\ &+ B \sum_{j=1}^{k} v_j q_j(\mathbf{x}, \mathbf{y}), \end{aligned} \tag{8}$$

where B represents a tunable scaling factor, and $\mathbf{f}_1$ and $\mathbf{f}_2$ represent the feature vectors of first and second-order parsing parts, respectively.

4 Experiments

We now describe experiments investigating the effectiveness of the SS-SCM approach for dependency parsing. The experiments test basic, first-order parsing models, as well as the extensions to cluster-based features and second-order parsing models described in the previous section.

4.1 Data Sets

We conducted experiments on both English and Czech data. We used the Wall Street Journal sections of the Penn Treebank (PTB) III (Marcus et al., 1994) as a source of labeled data for English, and the Prague Dependency Treebank (PDT) 1.0 (Hajič, 1998) for Czech. To facilitate comparisons with previous work, we used exactly the same training, development and test sets as those described in (McDonald et al., 2005a; McDonald et al., 2005b; McDonald and Pereira, 2006; Koo et al., 2008). The English dependency-parsing data sets were constructed using a standard set of head-selection rules (Yamada and Matsumoto, 2003) to convert the phrase structure syntax of the Treebank to dependency tree representations. We split the data into three parts: sections 02-21 for training, section 22 for development and section 23 for test. The Czech data sets were obtained from the predefined training/development/test partition in the PDT. The unlabeled data for English was derived from the Brown Laboratory for Linguistic Information Processing (BLLIP) Corpus (LDC2000T43)[2], giving a total of 1,796,379 sentences and 43,380,315 tokens. The raw text section of the PDT was used for Czech, giving 2,349,224 sentences and 39,336,570 tokens. These data sets are identical to the unlabeled data used in (Koo et al., 2008), and are disjoint from the training, development and test sets. The datasets used in our experiments are summarized in Table 1.

In addition, we will describe experiments that make use of much larger amounts of unlabeled data. Unfortunately, we have no data available other than PDT for Czech, this is done only for English dependency parsing. Table 2 shows the detail of the larger unlabeled data set used in our experiments, where we eliminated sentences that have more than 128 tokens for computational reasons. Note that the total size of the unlabeled data reaches 3.72G (billion) tokens, which is approxi-

[2]We ensured that the sentences used in the PTB were excluded from the unlabeled data, since sentences used in BLLIP corpus are a super-set of the PTB.

mately 4,000 times larger than the size of labeled training data.

4.2 Features

4.2.1 Baseline Features

In general we will assume that the input sentences include both words and part-of-speech (POS) tags. Our baseline features ("baseline") are very similar to those described in (McDonald et al., 2005a; Koo et al., 2008): these features track word and POS bigrams, contextual features surrounding dependencies, distance features, and so on. English POS tags were assigned by MXPOST (Ratnaparkhi, 1996), which was trained on the training data described in Section 4.1. Czech POS tags were obtained by the following two steps: First, we used 'feature-based tagger' included with the PDT[3], and then, we used the method described in (Collins et al., 1999) to convert the assigned rich POS tags into simplified POS tags.

4.2.2 Cluster-based Features

In a second set of experiments, we make use of the feature set used in the semi-supervised approach of (Koo et al., 2008). We will refer to this as the "cluster-based feature set" (CL). The BLLIP (43M tokens) and PDT (39M tokens) unlabeled data sets shown in Table 1 were used to construct the hierarchical clusterings used within the approach. Note that when this feature set is used within the SS-SCM approach, the same set of unlabeled data is used to both induce the clusters, and to estimate the generative models within the SS-SCM model.

4.2.3 Constructing the Generative Models

As described in section 2.2, the generative models in the SS-SCM approach are defined through a partition of the original feature vector $\mathbf{f}(\mathbf{x}, \mathbf{y})$ into k feature vectors $\mathbf{r}_1(\mathbf{x}, \mathbf{y}) \ldots \mathbf{r}_k(\mathbf{x}, \mathbf{y})$. We follow a similar approach to that of (Suzuki and Isozaki, 2008) in partitioning $\mathbf{f}(\mathbf{x}, \mathbf{y})$, where the k different feature vectors correspond to different feature types or feature templates. Note that, in general, we are not necessary to do as above, this is one systematic way of a feature design for this approach.

4.3 Other Experimental Settings

All results presented in our experiments are given in terms of parent-prediction accuracy on *unlabeled* dependency parsing. We ignore the parent-predictions of punctuation tokens for English, while we retain all the punctuation tokens for Czech. These settings match the evaluation setting in previous work such as (McDonald et al., 2005a; Koo et al., 2008).

We used the method proposed by (Carreras, 2007) for our second-order parsing model. Since this method only considers projective dependency structures, we "projectivized" the PDT training data in the same way as (Koo et al., 2008). We used a non-projective model, trained using an application of the matrix-tree theorem (Koo et al., 2007; Smith and Smith, 2007; McDonald and Satta, 2007) for the first-order Czech models, and projective parsers for all other models.

As shown in Section 2, SS-SCMs with 1st-order parsing models have two tunable parameters, C and η, corresponding to the regularization constant, and the Dirichlet prior for the generative models. We selected a fixed value $\eta = 2$, which was found to work well in preliminary experiments.[4] The value of C was chosen to optimize performance on development data. Note that C for supervised SCMs were also tuned on development data. For the two-stage SS-SCM for incorporating second-order parsing model, we have additional one tunable parameter B shown in Eq. 8. This was also chosen by the value that provided the best performance on development data.

In addition to providing results for models trained on the full training sets, we also performed experiments with smaller labeled training sets. These training sets were either created through random sampling or by using a predefined subset of document IDs from the labeled training data.

5 Results and Discussion

Table 3 gives results for the SS-SCM method under various configurations: for first and second-order parsing models, with and without the cluster features of (Koo et al., 2008), and for varying amounts of labeled data. The remainder of this section discusses these results in more detail.

5.1 Effects of the Quantity of Labeled Data

We can see from the results in Table 3 that our semi-supervised approach consistently gives gains

[3] Training, development, and test data in PDT already contains POS tags assigned by the 'feature-based tagger'.

[4] An intuitive meaning of $\eta = 2$ is that this adds one pseudo expected count to every feature when estimating new parameter values.

(a) English dependency parsing: w/ 43M token unlabeled data (BLLIP)

WSJ sec. IDs		wsj_21		random selection		random selection		wsj_15–18		wsj_02-21(all)	
# of sentences / tokens		1,671 / 40,039		2,000 / 48,577		8,000 / 190,958		8,936 / 211,727		39,832 / 950,028	
feature type		baseline	CL	baseline	CL	baseline	CL	baseline	CL	baseline	CL
Supervised SCM	(1od)	85.63	86.80	87.02	88.05	89.23	90.45	89.43	90.85	91.21	92.53
SS-SCM	(1od)	87.16	88.40	88.07	89.55	90.06	91.45	90.23	91.63	91.72	93.01
(gain over Sup. SCM)		(+1.53)	(+1.60)	(+1.05)	(+1.50)	(+0.83)	(+1.00)	(+0.80)	(+0.78)	(+0.51)	(+0.48)
Supervised MIRA	(2od)	87.99	89.05	89.20	90.06	91.20	91.75	91.50	92.14	93.02	93.54
2-stage SS-SCM(+MIRA)	(2od)	88.88	89.94	90.03	90.90	91.73	92.51	91.95	92.73	93.45	94.13
(gain over Sup. MIRA)		(+0.89)	(+0.89)	(+0.83)	(+0.84)	(+0.53)	(+0.76)	(+0.45)	(+0.59)	(+0.43)	(+0.59)

(b) Czech dependency parsing: w/ 39M token unlabeled data (PDT)

PDT Doc. IDs		random selection		c[0-9]*		random selection		l[a-i]*		(all)	
# of sentences / tokens		2,000 / 34,722		3,526 / 53,982		8,000 / 140,423		14,891 / 261,545		73,008 /1,225,590	
feature type		baseline	CL	baseline	CL	baseline	CL	baseline	CL	baseline	CL
Supervised SCM	(1od)	75.67	77.82	76.88	79.24	80.61	82.85	81.94	84.47	84.43	86.72
SS-SCM	(1od)	76.47	78.96	77.61	80.28	81.30	83.49	82.74	84.91	85.00	87.03
(gain over Sup. SCM)		(+0.80)	(+1.14)	(+0.73)	(+1.04)	(+0.69)	(+0.64)	(+0.80)	(+0.44)	(+0.57)	(+0.31)
Supervised MIRA	(2od)	78.19	79.60	79.58	80.77	83.15	84.39	84.27	85.75	86.82	87.76
2-stage SS-SCM(+MIRA)	(2od)	78.71	80.09	80.37	81.40	83.61	84.87	84.95	86.00	87.03	88.03
(gain over Sup. MIRA)		(+0.52)	(+0.49)	(+0.79)	(+0.63)	(+0.46)	(+0.48)	(+0.68)	(+0.25)	(+0.21)	(+0.27)

Table 3: Dependency parsing results for the SS-SCM method with different amounts of labeled training data. Supervised SCM (1od) and Supervised MIRA (2od) are the baseline first and second-order approaches; SS-SCM (1od) and 2-stage SS-SCM(+MIRA) (2od) are the first and second-order approaches described in this paper. "Baseline" refers to models without cluster-based features, "CL" refers to models which make use of cluster-based features.

in performance under various sizes of labeled data. Note that the baseline methods that we have used in these experiments are strong baselines. It is clear that the gains from our method are larger for smaller labeled data sizes, a tendency that was also observed in (Koo et al., 2008).

5.2 Impact of Combining SS-SCM with Cluster Features

One important observation from the results in Table 3 is that SS-SCMs can successfully improve the performance over a baseline method that uses the cluster-based feature set (CL). This is in spite of the fact that the generative models within the SS-SCM approach were trained on the same unlabeled data used to induce the cluster-based features.

5.3 Impact of the Two-stage Approach

Table 3 also shows the effectiveness of the two-stage approach (described in Section 3.2) that integrates the SS-SCM method within a second-order parser. This suggests that the SS-SCM method can be effective in providing features (generative models) used within a separate learning algorithm, providing that this algorithm can make use of real-valued features.

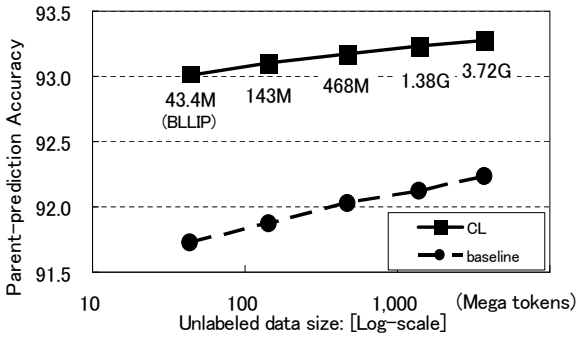


Figure 1: Impact of unlabeled data size for the SS-SCM on development data of English dependency parsing.

5.4 Impact of the Amount of Unlabeled Data

Figure 1 shows the dependency parsing accuracy on English as a function of the amount of unlabeled data used within the SS-SCM approach. (As described in Section 4.1, we have no unlabeled data other than PDT for Czech, hence this section only considers English dependency parsing.) We can see that performance does improve as more unlabeled data is added; this trend is seen both with and without cluster-based features. In addition, Table 4 shows the performance of our proposed method using 3.72 billion tokens of unla-

feature type		baseline	CL
SS-SCM	(1st-order)	92.23	93.23
(gain over Sup. SCM)		(+1.02)	(+0.70)
2-stage SS-SCM(+MIRA)	(2nd-order)	93.68	94.26
(gain over Sup. MIRA)		(+0.66)	(+0.72)

Table 4: Parent-prediction accuracies on development data with 3.72G tokens unlabeled data for English dependency parsing.

beled data. Note, however, that the gain in performance as unlabeled data is added is not as sharp as might be hoped, with a relatively modest difference in performance for 43.4 million tokens vs. 3.72 billion tokens of unlabeled data.

5.5 Computational Efficiency

The main computational challenge in our approach is the estimation of the generative models $\mathbf{q} = \langle q_1 \dots q_k \rangle$ from unlabeled data, particularly when the amount of unlabeled data used is large. In our implementation, on the 43M token BLLIP corpus, using baseline features, it takes about 5 hours to compute the expected counts required to estimate the parameters of the generative models on a single 2.93GHz Xeon processor. It takes roughly 18 days of computation to estimate the generative models from the larger (3.72 billion word) corpus. Fortunately it is simple to parallelize this step; our method takes a few hours on the larger data set when parallelized across around 300 separate processes.

Note that once the generative models have been estimated, decoding with the model, or training the model on labeled data, is relatively inexpensive, essentially taking the same amount of computation as standard dependency-parsing approaches.

5.6 Results on Test Data

Finally, Table 5 displays the final results on test data. There results are obtained using the best setting in terms of the development data performance. Note that the English dependency parsing results shown in the table were achieved using 3.72 billion tokens of unlabeled data. The improvements on test data are similar to those observed on the development data. To determine statistical significance, we tested the difference of parent-prediction error-rates at the sentence level using a paired Wilcoxon signed rank test. All eight comparisons shown in Table 5 are significant with $p < 0.01$.

(a) English dependency parsing: w/ 3.72G token ULD

feature set		baseline	CL
SS-SCM	(1st-order)	91.89	92.70
(gain over Sup. SCM)		(+0.92)	(+0.58)
2-stage SS-SCM(+MIRA)	(2nd-order)	93.41	93.79
(gain over Sup. MIRA)		(+0.65)	(+0.48)

(b) Czech dependency parsing: w/ 39M token ULD (PDT)

feature set		baseline	CL
SS-SCM	(1st-order)	84.98	87.14
(gain over Sup. SCM)		(+0.58)	(+0.39)
2-stage SS-SCM(+MIRA)	(2nd-order)	86.90	88.05
(gain over Sup. MIRA)		(+0.15)	(+0.36)

Table 5: Parent-prediction accuracies on test data using the best setting in terms of development data performances in each condition.

(a) English dependency parsers on PTB

dependency parser	test	description
(McDonald et al., 2005a)	90.9	1od
(McDonald and Pereira, 2006)	91.5	2od
(Koo et al., 2008)	92.23	1od, 43M ULD
SS-SCM (w/ CL)	**92.70**	1od, 3.72G ULD
(Koo et al., 2008)	93.16	2od, 43M ULD
2-stage SS-SCM(+MIRA, w/ CL)	**93.79**	2od, 3.72G ULD

(b) Czech dependency parsers on PDT

dependency parser	test	description
(McDonald et al., 2005b)	84.4	1od
(McDonald and Pereira, 2006)	85.2	2od
(Koo et al., 2008)	86.07	1od, 39M ULD
(Koo et al., 2008)	87.13	2od, 39M ULD
SS-SCM (w/ CL)	**87.14**	1od, 39M ULD
2-stage SS-SCM(+MIRA, w/ CL)	**88.05**	2od, 39M ULD

Table 6: Comparisons with the previous top systems: (1od, 2od: 1st- and 2nd-order parsing model, ULD: unlabeled data).

6 Comparison with Previous Methods

Table 6 shows the performance of a number of state-of-the-art approaches on the English and Czech data sets. For both languages our approach gives the best reported figures on these datasets. Our results yield relative error reductions of roughly 27% (English) and 20% (Czech) over McDonald and Pereira (2006)'s second-order supervised dependency parsers, and roughly 9% (English) and 7% (Czech) over the previous best results provided by Koo et. al. (2008)'s second-order semi-supervised dependency parsers.

Note that there are some similarities between our two-stage semi-supervised learning approach and the semi-supervised learning method introduced by (Blitzer et al., 2006), which is an extension of the method described by (Ando and Zhang,

2005). In particular, both methods use a two-stage approach; They first train generative models or auxiliary problems from unlabeled data, and then, they incorporate these trained models into a supervised learning algorithm as real valued features. Moreover, both methods make direct use of existing feature-vector definitions $\mathbf{f}(\mathbf{x}, \mathbf{y})$ in inducing representations from unlabeled data.

7 Conclusion

This paper has described an extension of the semi-supervised learning approach of (Suzuki and Isozaki, 2008) to the dependency parsing problem. In addition, we have described extensions that incorporate the cluster-based features of Koo et al. (2008), and that allow the use of second-order parsing models. We have described experiments that show that the approach gives significant improvements over state-of-the-art methods for dependency parsing; performance improves when the amount of unlabeled data is increased from 43.8 million tokens to 3.72 billion tokens. The approach should be relatively easily applied to languages other than English or Czech.

We stress that the SS-SCM approach requires relatively little hand-engineering: it makes direct use of the existing feature-vector representation $\mathbf{f}(\mathbf{x}, \mathbf{y})$ used in a discriminative model, and does not require the design of new features. The main choice in the approach is the partitioning of $\mathbf{f}(\mathbf{x}, \mathbf{y})$ into components $\mathbf{r}_1(\mathbf{x}, \mathbf{y}) \ldots \mathbf{r}_k(\mathbf{x}, \mathbf{y})$, which in our experience is straightforward.

References

R. Kubota Ando and T. Zhang. 2005. A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *Journal of Machine Learning Research*, 6:1817–1853.

J. K. Baker. 1979. Trainable Grammars for Speech Recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550.

J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain Adaptation with Structural Correspondence Learning. In *Proc. of EMNLP-2006*, pages 120–128.

P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. 1992. Class-based n-gram Models of Natural Language. *Computational Linguistics*, 18(4):467–479.

S. Buchholz and E. Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proc. of CoNLL-X*, pages 149–164.

X. Carreras. 2007. Experiments with a Higher-Order Projective Dependency Parser. In *Proc. of EMNLP-CoNLL*, pages 957–961.

M. Collins, J. Hajic, L. Ramshaw, and C. Tillmann. 1999. A Statistical Parser for Czech. In *Proc. of ACL*, pages 505–512.

J. Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proc. of COLING-96*, pages 340–345.

Jan Hajič. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, pages 12–19. Prague Karolinum, Charles University Press.

T. Koo, A. Globerson, X. Carreras, and M. Collins. 2007. Structured Prediction Models via the Matrix-Tree Theorem. In *Proc. of EMNLP-CoNLL*, pages 141–150.

T. Koo, X. Carreras, and M. Collins. 2008. Simple Semi-supervised Dependency Parsing. In *Proc. of ACL-08: HLT*, pages 595–603.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. of ICML-2001*, pages 282–289.

D. C. Liu and J. Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *Math. Programming, Ser. B*, 45(3):503–528.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

R. McDonald and F. Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proc. of EACL*, pages 81–88.

R. McDonald and G. Satta. 2007. On the Complexity of Non-Projective Data-Driven Dependency Parsing. In *Proc. of IWPT*, pages 121–132.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online Large-margin Training of Dependency Parsers. In *Proc. of ACL*, pages 91–98.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005b. Non-projective Dependency Parsing using Spanning Tree Algorithms. In *Proc. of HLT-EMNLP*, pages 523–530.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proc. of EMNLP-CoNLL*, pages 915–932.

Mark A. Paskin. 2001. Cubic-time Parsing and Learning Algorithms for Grammatical Bigram. Technical report, University of California at Berkeley, Berkeley, CA, USA.

A. Ratnaparkhi. 1996. A Maximum Entropy Model for Part-of-Speech Tagging. In *Proc. of EMNLP*, pages 133–142.

D. A. Smith and J. Eisner. 2007. Bootstrapping Feature-Rich Dependency Parsers with Entropic Priors. In *Proc. of EMNLP-CoNLL*, pages 667–677.

D. A. Smith and N. A. Smith. 2007. Probabilistic Models of Nonprojective Dependency Trees. In *Proc. of EMNLP-CoNLL*, pages 132–140.

J. Suzuki and H. Isozaki. 2008. Semi-supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data. In *Proc. of ACL-08: HLT*, pages 665–673.

Q. I. Wang, D. Schuurmans, and D. Lin. 2008. Semi-supervised Convex Training for Dependency Parsing. In *Proc. of ACL-08: HLT*, pages 532–540.

H. Yamada and Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proc. of IWPT*.

Statistical Bistratal Dependency Parsing

Richard Johansson
Department of Information Engineering and Computer Science
University of Trento
Trento, Italy
johansson@disi.unitn.it

Abstract

We present an inexact search algorithm for the problem of predicting a two-layered dependency graph. The algorithm is based on a k-best version of the standard cubic-time search algorithm for projective dependency parsing, which is used as the backbone of a beam search procedure. This allows us to handle the complex non-local feature dependencies occurring in bistratal parsing if we model the interdependency between the two layers.

We apply the algorithm to the syntactic–semantic dependency parsing task of the CoNLL-2008 Shared Task, and we obtain a competitive result equal to the highest published for a system that jointly learns syntactic and semantic structure.

1 Introduction

Numerous linguistic theories assume a multistratal model of linguistic structure, such as a layer of surface syntax, deep syntax, and shallow semantics. Examples include Meaning–Text Theory (Mel'čuk, 1988), Discontinuous Grammar (Buch-Kromann, 2006), Extensible Dependency Grammar (Debusmann et al., 2004), and the Functional Generative Description (Sgall et al., 1986) which forms the theoretical foundation of the Prague Dependency Treebank (Hajič, 1998).

In the statistical NLP community, the most widely used grammatical resource is the Penn Treebank (Marcus et al., 1993). This is a purely syntactic resource, but we can also include this treebank in the category of multistratal resources since the PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004) projects have annotated shallow semantic structures on top of it. Dependency-converted versions of the Penn Treebank, PropBank and NomBank were used in the CoNLL-2008 Shared Task (Surdeanu et al., 2008), in which the task of the participants was to produce a bistratal dependency structure consisting of surface syntax and shallow semantics.

Producing a consistent multistratal structure is a conceptually and computationally complex task, and most previous methods have employed a purely pipeline-based decomposition of the task. This includes the majority of work on shallow semantic analysis (Gildea and Jurafsky, 2002, *inter alia*). Nevertheless, since it is obvious that syntax and semantics are highly interdependent, it has repeatedly been suggested that the problems of syntactic and semantic analysis should be carried out *simultaneously* rather than in a pipeline, and that modeling the interdependency between syntax and semantics would improve the quality of all the substructures.

The purpose of the CoNLL-2008 Shared Task was to study the feasibility of a joint analysis of syntax and semantics, and while most participating systems used a pipeline-based approach to the problem, there were a number of contributions that attempted to take the interdependence between syntax and semantics into account. The top-performing system in the task (Johansson and Nugues, 2008) applied a very simple reranking scheme by means of a k-best syntactic output, similar to previous attempts (Gildea and Jurafsky, 2002; Toutanova et al., 2005) to improve semantic role labeling performance by using mul-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 561–569,
Singapore, 6-7 August 2009.

tiple parses. The system by Henderson et al. (2008) extended previous stack-based algorithms for dependency parsing by using two separate stacks to build the syntactic and semantic graphs. Lluís and Màrquez (2008) proposed a model that simultaneously predicts syntactic and semantic links, but since its search algorithm could not take the syntactic–semantic interdependencies into account, a pre-parsing step was still needed. In addition, before the CoNLL-2008 shared task there have been a few attempts to jointly learn syntactic and semantic structure; for instance, Merlo and Musillo (2008) appended semantic role labels to the phrase tags in a constituent treebank and applied a conventional constituent parser to predict constituent structure and semantic roles.

In this paper, we propose a new approximate search method for bistratal dependency analysis. The search method is based on a beam search procedure that extends a k-best version of the standard cubic-time search algorithm for projective dependency parsing. This is similar to the search method for constituent parsing used by Huang (2008), who referred to it as *cube pruning*, inspired by an idea from machine translation decoding (Chiang, 2007). The cube pruning approach, which is normally used to solve the $\arg\max$ problem, was also recently extended to summing problems, which is needed in some learning algorithms (Gimpel and Smith, 2009).

We apply the algorithm on the CoNLL-2008 Shared Task data, and obtain the same evaluation score as the best previously published system that simultaneously learns syntactic and semantic structure (Titov et al., 2009).

2 Bistratal Dependency Parsing

In the tradition of dependency representation of sentence structure, starting from Tesnière (1959), the linguistic structure of the sentence is represented as a directed graph of relations between words. In most theories, certain constraints are imposed on this graph; the most common constraint on dependency graphs in syntax, for instance, is that the graph should form a tree (i.e. it should be connected, acyclic, and every node should have at most one incoming edge). This assumption underlies almost all dependency parsing, although there are also a few parsers based on slightly more general problem formulations (Sagae and Tsuji, 2008).

In this paper, we assume a different type of constraint: that the graph can be partitioned into two subgraphs that we will refer to as *strata* or *layers*, where the first of the layers forms a tree. For the second layer, the only assumption we make is that there is at most one link between any two words. However, we believe that for any interesting linguistic structure, the second layer will be highly dependent on the structure of the first layer.

Figure 1 shows an example of a bistratal dependency graph such as in the CoNLL-2008 Shared Task on syntactic and semantic dependency parsing. The figure shows the representation of the sentence *We were expecting prices to fall*. The primary layer represents surface-syntactic relations, shown above the sentence, and the secondary layer consists of predicate–argument links (here, we have two predicates *expecting* and *fall*).

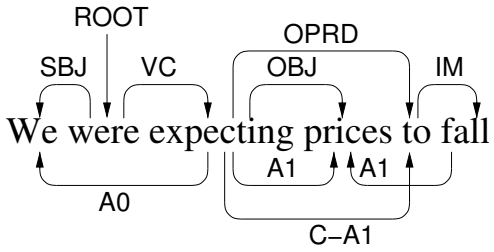


Figure 1: Example of a bistratal dependency graph.

We now give a formal model of the statistical parsing problem of prediction of a bistratal dependency graph. For a given input sentence $\boldsymbol{x}$, the task of our algorithm is to predict a structure $\hat{y}$ consisting of a *primary layer* $\hat{y}_p$ and a *secondary layer* $\hat{y}_s$. In a discriminative modeling framework, we model this prediction problem as the search for the highest-scoring output from the candidate space $\mathcal{Y}$ under a scoring function F:

$$\langle \hat{y}_p, \hat{y}_s \rangle = \arg\max_{\langle y_p, y_s \rangle \in \mathcal{Y}} F(\boldsymbol{x}, y_p, y_s)$$

The learning problem consists of searching in the model space for a scoring function F that minimizes the cost of predictions on unseen examples according to a given *cost function* ρ. In this work, we consider linear scoring functions of the following form:

$$F(\boldsymbol{x}, y_p, y_s) = \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, y_p, y_s)$$

where $\boldsymbol{\Phi}(\boldsymbol{x}, y)$ is a numeric feature representation of the tuple $(\boldsymbol{x}, y_p, y_s)$ and $\boldsymbol{w}$ a high-dimensional vector of feature weights.

Based on the structural assumptions made above, we now decompose the feature representation into three parts:

$$\mathbf{\Phi} = \mathbf{\Phi}_p + \mathbf{\Phi}_i + \mathbf{\Phi}_s$$

Here, $\mathbf{\Phi}_p$ represents the primary layer, assumed to be a tree, $\mathbf{\Phi}_s$ the secondary layer, and finally $\mathbf{\Phi}_i$ is the representation of the interdependency between the layers. For the feature representations of the primary and secondary layers, we employ *edge factorization*, a decomposition widely used in statistical dependency parsing, and assume that all edges can be scored independently:

$$\mathbf{\Phi}_p(\boldsymbol{x}, y_p) = \sum_{f \in y_p} \phi_p(\boldsymbol{x}, f)$$

The representation of the interdependency between the layers assumes that each secondary link is dependent on the primary layer, but independent of other secondary links.

$$\mathbf{\Phi}_i(\boldsymbol{x}, y_p, y_s) = \sum_{f \in y_s} \phi_i(\boldsymbol{x}, f, y_p)$$

The interdependency between layers is the bottleneck for the search algorithm that we will present in Section 3. For semantic role analysis, this involves all features that rely on a syntactic representation, most importantly the PATH feature that represents the grammatical relation between predicate and argument words. For instance, in Figure 1, we can represent the surface-syntactic relation between the tokens *fall* and *prices* as the string IM↑OPRD↑OBJ↓. In this work, all interdependency features will be based on paths in the primary layer.

3 A Bistratal Search Algorithm

This section presents an algorithm to approximately solve the $\arg\max$ problem for prediction of bistratal dependency structures. We present the algorithm in two steps: first, we review a k-best version of the standard search algorithm for projective monostratal dependency parsing, based on the work by Huang and Chiang (2005).[1] In the second step, starting from the k-best monostratal search, we devise a search method for the bistratal problem.

[1]Huang and Chiang (2005) described an even more efficient k-best algorithm based on lazy evaluation, which we will not use here since it is not obviously adaptable to the situation where the search is inexact.

3.1 Review of k-Best Dependency Parsing

The search method commonly used in dependency parsers is a chart-based dynamic programming algorithm that finds the highest-scoring projective dependency tree under an edge-factored scoring function. It runs in cubic time with respect to the sentence length. In a slightly more general formulation, it was first published by Eisner (1996). Starting from McDonald et al. (2005), it has been widely used in recent statistical dependency parsing frameworks.

The algorithm works by creating *open structures*, which consist of a dependency link and the set of links that it spans, and *closed structures*, consisting of the left or right half of a complete subtree. An open structure is created by a procedure LINK that adds a dependency link to connect a right-pointing and a left-pointing closed structure, and a closed structure by a procedure JOIN that joins an open structure with a closed structure. Figure 2 shows schematic illustrations: a LINK operation connects the right-pointing closed structure between s and j with the left-pointing closed structure between $j+1$ and e, and a JOIN operation connects an open structure between s and j with a closed structure between j and e.

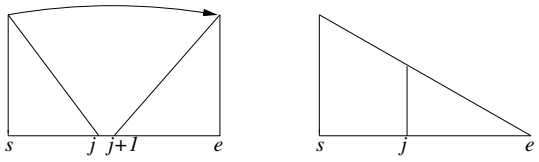


Figure 2: Illustrations of the LINK and JOIN operations.

The search algorithm can easily be extended to find the k best parses, not only the best one. In k-best parsing, we maintain a k-best list in every cell in the dynamic programming table. To create the k-best list of derivations for an open structure between the positions s and e, for instance, there are up to $|L| \cdot (e-s) \cdot k^2$ possible combinations to consider if the set of allowed labels is L. The key observation by Huang and Chiang (2005) is to make use of the fact that the lists are sorted. For every position between s and e, we add the best combination to a priority queue, from which we then repeatedly remove the front item. For every item we remove, we add three successors: an item with a next-best left part, an item with a next-best right part, and finally an item with a next-best edge

label.

The pseudocode of the search algorithm for k-best dependency parsing is given in Algorithms 1 and 2. For brevity, we omitted the code for ADVANCE-LEFT and ADVANCE-RIGHT, which are similar to ADVANCE-EDGE, as well as ADVANCE-LOWER, which resembles ADVANCE-UPPER. The FST function used in the pseudocode returns the first element of a tuple.

The algorithm uses a priority queue with standard operations ENQUEUE, which enqueues an element, and DEQUEUE, which removes the highest-scoring item from the queue. With a standard binary heap implementation of the priority queue, these two operations execute in logarithmic time. To build the queue, we use a constant-time TOSS operation, which appends an item to the queue without enforcing the priority queue constraint, and a HEAPIFY operation that constructs a consistent priority queue in linear time.

3.2 Extension to Bistratal Dependency Parsing

The k-best algorithm forms the core of the inexact bistratal search algorithm. Our method is similar to the forest reranking method by Huang (2008), although there is no forest pruning or reranking involved here. Crucially, we divide the features into *local* features, which can be computed "offline", and *nonlocal* features, which must be computed during search. In our case, the local features are $\mathbf{\Phi}_p$ and $\mathbf{\Phi}_s$, while the nonlocal features are the interdependent features $\mathbf{\Phi}_i$.

Algorithm 3 shows pseudocode for the main part of the bistratal search algorithm, and Algorithm 4 for its support functions. The algorithm works as follows: for every span $\langle s, e \rangle$, the algorithm first uses the LINK procedure from the k-best monostratal search to construct a k-best list of open structures without semantic links. In the next step, secondary links are added in the procedure LINK-SECONDARY. For brevity, we show only the procedures that create open structures; they are very similar to their closed-structure counterparts.

The LINK-SECONDARY procedure starts by creating an initial candidate (FIRST-SEC-OPEN) based on the best open structure for the primary layer. FIRST-SEC-OPEN creates the candidate space for secondary links for a single primary open structure. To reduce search complexity, it makes use of a problem-specific function SCOPE that defines which secondary links are possible from a given token, given a primary-layer context.

An important insight by Huang (2008) is that nonlocal features should be computed as early as possible during search. In our case, we assume that the interdependency features are based on *tree paths* in the primary layer. This means that secondary links between two tokens can be added when there is a complete path in the primary layer between the tokens. When we create an open

Algorithm 1 k-best search algorithm for dependency parsing.

```
function k-BEST-SEARCH(k)
  n ← length of the sentence
  initialize the table O of open structures
  initialize the table C of closed structures
  for m ∈ [1, ..., n]
    for s ∈ [0, ..., n − m]
      LINK(s, s + m, →, k)
      LINK(s, s + m, ←, k)
      JOIN(s, s + m, →, k)
      JOIN(s, s + m, ←, k)
  return C[0, n, →]

procedure LINK(s, e, dir, k)
  E ← CREATE-EDGES(s, e, dir, k)
  q ← empty priority queue
  for j ∈ [s, ..., e − 1]
    l ← C[s, j, →]
    r ← C[j + 1, e, ←]
    o ← CREATE-OPEN(E, l, r, 1, 1, 1)
    TOSS(q, o)
  HEAPIFY(q)
  while |O[s, e, dir]| < k and |q| > 0
    o ← DEQUEUE(q)
    if o ∉ O[s, e, dir]
      APPEND(O[s, e, dir], o)
      ENQUEUE(q, ADVANCE-EDGE(o))
      ENQUEUE(q, ADVANCE-LEFT(o))
      ENQUEUE(q, ADVANCE-RIGHT(o))

procedure JOIN(s, e, dir, k)
  q ← empty priority queue
  if dir =→
    for j ∈ [s + 1, ..., e]
      u ← O[s, j, →]
      l ← C[j, e, →]
      c ← CREATE-CLOSED(u, l, 1, 1)
      TOSS(q, c)
  else
    for j ∈ [s, ..., e − 1]
      u ← O[j, e, ←]
      l ← C[s, j, ←]
      c ← CREATE-CLOSED(u, l, 1, 1)
      TOSS(q, c)
  HEAPIFY(q)
  while |C[s, e, dir]| < k and |q| > 0
    c ← DEQUEUE(q)
    if c ∉ C[s, e, dir]
      APPEND(C[s, e, dir], c)
      ENQUEUE(q, ADVANCE-UPPER(c))
      ENQUEUE(q, ADVANCE-LOWER(c))
```

Algorithm 2 Support operations for the k-best search.

function CREATE-EDGES(s, e, dir, k)
  $E \leftarrow \emptyset$
  for $l \in$ ALLOWED-LABELS(s, e, dir)
    $score_L \leftarrow \boldsymbol{w} \cdot \phi_p(s, e, dir, l)$
    $edge \leftarrow \langle score_L, s, e, dir, l \rangle$
    APPEND$(E, edge)$
  return the top k edges in E

function CREATE-OPEN(E, l, r, i_e, i_l, i_r)
  $score_L \leftarrow$ FST$(E[i_e])$ + FST$(l[i_l])$ + FST$(r[i_r])$
  return $\langle score_L + score_N, E, l, r, i_e, i_l, i_r \rangle$

function CREATE-CLOSED(u, l, i_u, i_r)
  $score_L \leftarrow$ FST$(u[i_u])$ + FST$(l[i_l])$
  return $\langle score_L + score_N, u, l, i_u, i_l \rangle$

function ADVANCE-EDGE(o)
where $o = (score, E, l, r, i_e, i_l, i_r)$
  if $i_e =$ LENGTH(E)
    return $\emptyset$
  else
    return CREATE-OPEN$(E, l, r, i_e + 1, i_l, i_r)$

function ADVANCE-UPPER(c)
where $c = (u, l, i_u, i_l)$
  if $i_u =$ LENGTH(u)
    return $\emptyset$
  else
    return CREATE-CLOSED$(u, l, i_u + 1, i_l)$

structure by adding a link between two substructures, a complete path is created between the tokens in the substructures. We thus search for possible secondary links only between the two substructures that are joined.

Figure 3 illustrates this process. A primary open structure between s and e has been created by adding a link from the right-pointing closed structure between s and j to the left-pointing closed structure between $j + 1$ and e. We now try to add secondary links between the two substructures. For instance, in the semantic role parsing task described in subsection 3.3, if we know that there is a predicate between s and j, then we look for arguments between $j + 1$ and e, i.e. we apply the SCOPE function to the right substructure.

When computing the scores for secondary links, note that for efficiency only the interdependent part $\boldsymbol{\Phi}_i$ should be computed in CREATE-SECEDGES; the part of the score that does not depend on the primary layer can be computed before entering the search procedure.

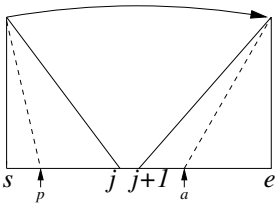


Figure 3: Illustration of the secondary linking process: When two substructures are connected, we can compute the path between a predicate in the left substructure and an argument in the right substructure.

Algorithm 3 Search algorithm for bistratal dependency parsing.

function BISTRATAL-SEARCH(k)
  $n \leftarrow$ length of the sentence
  initialize the table O of open structures
  initialize the table C of closed structures
  using ϕ_s, compute a table $scores_s$ for all
    possible secondary edges $\langle h, d, l \rangle$
  for $m \in [1, \ldots, n]$
    for $s \in [0, \ldots, n - m]$
      LINK$(s, s + m, \rightarrow, k)$
      LINK-SECONDARY$(s, s + m, \rightarrow, k)$
      LINK$(s, s + m, \leftarrow, k)$
      LINK-SECONDARY$(s, s + m, \leftarrow, k)$
      JOIN$(s, s + m, \rightarrow, k)$
      JOIN-SECONDARY$(s, s + m, \rightarrow, k)$
      JOIN$(s, s + m, \leftarrow, k)$
      JOIN-SECONDARY$(s, s + m, \leftarrow, k)$
  return FIRST$(C[0, n, \rightarrow])$

procedure LINK-SECONDARY(s, e, dir, k)
  $q \leftarrow$ empty priority queue
  $o \leftarrow$ FIRST-SEC-OPEN$(O[s, e, dir], 1, k)$
  ENQUEUE(q, o)
  $buf \leftarrow$ empty list
  while $|buf| < k$ and $|q| > 0$
    $o \leftarrow$ DEQUEUE(q)
    if $o \notin buf$
      APPEND(buf, o)
      for $o' \in$ ADVANCE-SEC-OPEN(o, k)
        ENQUEUE(q, o')
  SORT(buf) to $O[s, e, dir]$

3.3 Application on the CoNLL-2008 Shared Task Treebank

We applied the bistratal search method in Algorithm 3 on the data from the CoNLL-2008 Shared Task (Surdeanu et al., 2008). Here, the primary layer is the tree of surface-syntactic relations such as subject and object, and the secondary layer contains the links between the predicate words in the sentence and their respective logical arguments, such as agent and patient. The training corpus consists of sections 02 – 21 of the Penn Treebank, and contains roughly 1 million words.

Algorithm 4 Support operations in bistratal search.

function FIRST-SEC-OPEN(L, i_L, k)
  if $i = \text{LENGTH}(L)$
    return $\emptyset$
  $l \leftarrow$GET-LEFT($L[i_L]$), $r \leftarrow$GET-RIGHT($L[i_L]$)
  for $h \in [\text{START}(l), \ldots, \text{END}(l)]$
    for $d \in \text{SCOPE}(r, h)]$
      $E[h][d] \leftarrow \text{CREATE-SEC-EDGES}(h, d, L[i_L], k)]$
      $I_E[h][d] \leftarrow 1$
  for $h \in [\text{START}(r), \ldots, \text{END}(r)]$
    for $d \in \text{SCOPE}(l, h)]$
      $E[h][d] \leftarrow \text{CREATE-SEC-EDGES}(h, d, L[i_L], k)]$
      $I_E[h][d] \leftarrow 1$
  return CREATE-SEC-OPEN(L, i_L, E, I)

function CREATE-SEC-EDGES(h, d, o, k)
  $E \leftarrow \emptyset$
  for $l \in$ ALLOWED-SEC-LABELS(h, d)
    $score \leftarrow \boldsymbol{w} \cdot \phi_i(h, d, l, o) + scores_s[h, d, l]$
    $edge \leftarrow \langle score, h, d, l \rangle$
    APPEND($E, edge$)
  return the top k edges in E

function CREATE-SEC-OPEN(L, i_L, E, I)
  $score \leftarrow \text{FST}(L[i_L]) + \sum_{h,d} \text{FST}(E[h, d, I_E[h, d]])$
  return $\langle score, L, i_L, E, I_E \rangle$

function ADVANCE-SEC-OPEN(o, k)
where $o = \langle score, L, i_L, E, I_E \rangle$
  $buf \leftarrow \emptyset$
  if $i_L <$ LENGTH(L) and $I_E = [1, \ldots, 1]$
    APPEND(buf, FIRST-SEC-OPEN($L, i_L + 1, k$))
  for h, d
    if $I_E[h, d] <$ LENGTH($E[h, d]$)
      $I'_E \leftarrow \text{COPY}(I_E)$
      $I'_E[h, d] \leftarrow I'_E[h, d] + 1$
      APPEND(buf, CREATE-SEC-OPEN(L, i_L, E, I'_E))
  return buf

To apply the bistratal search algorithm to the problem of syntactic–semantic parsing, a problem-specific implementation of the SCOPE function is needed. In this case, we made two assumptions. First, we assumed that the identities of the predicate words are known a priori[2]. Secondly, we assumed that every argument of a given predicate word is either a direct dependent of the predicate, one of its ancestors, or a direct dependent of one of its ancestors. This assumption is a simple adaptation of the pruning algorithm by Xue and Palmer (2004), and it holds for the vast majority of arguments in the CoNLL-2008 data; in the training set, we measured that this covers 99.04% of the arguments of verbs and 97.55% of the arguments of nouns.

Figure 4 shows an example of how the SCOPE function works in our case. If a predicate is contained in the right substructure, we find two potential arguments: one at the start of the left substructure, and one more by recursively searching the left structure.

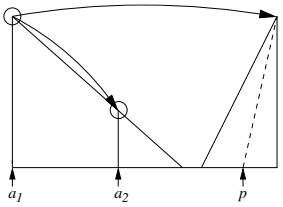


Figure 4: Illustration of the SCOPE function for predicate–argument links. If the right substructure contains a predicate, we can find potential arguments in the left substructure.

While the primary layer is assumed to be projective in Algorithm 3, the syntactic trees in the CoNLL-2008 data have a small number of nonprojective links. We used a pseudo-projective edge label encoding to handle nonprojectivity (Nivre and Nilsson, 2005).

To implement the model, we constructed feature representations $\boldsymbol{\Phi}_p$, $\boldsymbol{\Phi}_s$, and $\boldsymbol{\Phi}_i$. The surface-syntactic representation $\boldsymbol{\Phi}_p$ was a standard first-order edge factorization using the same features as McDonald et al. (2005). The features in $\boldsymbol{\Phi}_s$ and $\boldsymbol{\Phi}_i$ are shown in Table 1 and are standard features in statistical semantic role classification.

$\boldsymbol{\Phi}_s$	$\boldsymbol{\Phi}_i$
Predicate word	Path
Predicate POS	Path + arg. POS
Argument word	Path + pred. POS
Argument POS	Path + arg. word
Pred. + arg. words	Path + pred. word
Predicate word + label	Path + label
Predicate POS + label	Path + arg. POS + label
Argument word + label	Path + pred. POS + label
Argument POS + label	Path + arg. word + label
Pred. + arg. words + label	Path + pred. word + label

Table 1: Feature representation for secondary links.

We trained the discriminative model using the Online Passive–aggressive algorithm (Crammer et al., 2006), which is an efficient online learning method that can be used to train models for learning problems with structured output spaces. A cost function ρ is needed in the learning algorithm; we decomposed it into a pri-

[2]Since our algorithm needs to know the positions of the predicates, we trained a separate classifier using the LIBLINEAR toolkit (Fan et al., 2008) to identify the predicate words. As features for the classifier, we used the words and part-of-speech tags in a ± 3 window around the word under consideration.

mary part ρ_p and a secondary part ρ_s. We computed the primary part as the sum of link errors: $\rho_p(y_p, \hat{y}_p) = \sum_{l \in \hat{y}_p} c_p(l, y_p)$, where

$$c_p(l, y_p) = \begin{cases} 0 & \text{if } l \in y_p \text{ and its label is correct} \\ 0.5 & \text{if } l \in y_p \text{ but its label is incorrect} \\ 1 & \text{if } l \notin y_p \end{cases}$$

In a similar vein, we computed the secondary part ρ_s of the cost function as $\#fp + \#fn + 0.5 \cdot \#fl$, where $\#fp$ is the number of false positive secondary links, $\#fn$ the number of false negative links, and $\#fl$ the number of links with correct endpoints but incorrect label.

The training procedure took roughly 24 hours on an 2.3 GHz AMD Athlon processor. The memory consumption was about 1 GB during training.

4 Experiments

We evaluated the performance of our system on the test set from the CoNLL-2008 shared task, which consists of section 23 of the WSJ part of the Penn Treebank, as well as a small part of the Brown corpus. A beam width k of 4 was used in this experiment. Table 2 shows the results of the evaluation. The table shows the three most important scores computed by the official evaluation script: labeled syntactic dependency accuracy (LAS), labeled semantic dependency F_1-measure (Sem. F1), and the macro-averaged F_1-measure, a weighted combination of the syntactic and semantic scores (M. F1). Our result is competitive; we obtain the same macro F1 as the newly published result by Titov et al. (2009), which is the highest published figure for a joint syntactic–semantic parser so far. Importantly, our system clearly outperforms the system by Lluís and Màrquez (2008), which is the most similar system in problem modeling, but which uses a different search strategy.

System	LAS	Sem. F1	M. F1
This paper	86.6	77.1	81.8
Titov et al. (2009)	87.5	76.1	81.8
H. et al (2008)	87.6	73.1	80.5
L. & M. (2008)	85.8	70.3	78.1

Table 2: Results of published joint syntactic–semantic parsers on the CoNLL-2008 test set.

Since the search procedure is inexact, it is important to quantify roughly how much of a detrimental impact the approximation has on the parsing quality. We studied the influence of the beam width parameter k on the performance of the parser. The results on the development set can be seen in Table 3. As can be seen, a modest increase in performance can be obtained by increasing the beam width, at the cost of increased parsing time.

k	LAS	Sem. F1	M. F1	Time
1	85.14	77.05	81.10	242
2	85.43	77.17	81.30	369
4	85.49	77.20	81.35	625
8	85.58	77.20	81.40	1178

Table 3: Influence of beam width on parsing accuracy.

In addition, to have a rough indication of the impact of search errors on the quality of the parses, we computed the fraction of sentences where the gold-standard parse had a higher score according to the model than the parse returned by the search[3]. Table 4 shows the results of this experiment. This suggests that the search errors, although they clearly have an impact, are not the major source of errors, even with small beam widths.

k	Fraction
1	0.121
2	0.104
4	0.096
8	0.090

Table 4: Fraction of sentences in the development set where the gold-standard parse has a higher score than the parse returned by the search procedure.

To investigate where future optimization efforts should be spent, we used the built-in `hprof` profiling tool of Java to locate the bottlenecks. Once again, we ran the program on the development set with a beam width of 4, and Table 5 shows the three types of operations where the algorithm spent most of its time. It turns out that 74% of the time was spent on the computation and scoring of interdependency features. To make our algorithm truly useful in practice, we thus need to devise a way to speed up or cache these computations.

[3]To be able to compare the scores of the gold-standard and predicted parses, we disabled the automatic classifier for predicate identification and provided the parser with gold-standard predicates in this experiment.

Operation	Fraction
$\boldsymbol{w} \cdot \Phi_i$	0.64
Queue operations	0.15
Computation of Φ_i	0.10

Table 5: The three most significant bottlenecks and their fraction of the total runtime.

5 Discussion

In this paper, we have presented a new approximate search method to solve the problem of jointly predicting the two layers in a bistratal dependency graph. The algorithm shows competitive performance on the treebank used in the CoNLL-2008 Shared Task, a bistratal treebank consisting of a surface-syntactic and a shallow semantic layer. In addition to the syntactic–semantic task that we have described in this paper, we believe that our method can be used in other types of multistratal syntactic frameworks, such as a representation of surface and deep syntax as in Meaning–Text Theory (Mel'čuk, 1988).

The optimization problem that we set out to solve is intractable, but we have shown that reasonable performance can be achieved with an inexact, beam search-based search method. This is not obvious: it has previously been shown that using an inexact search procedure when the learning algorithm assumes that the search is exact may lead to slow convergence or even divergence (Kulesza and Pereira, 2008), but this does not seem to be a problem in our case.

While we used a beam search method as the method of approximation, other methods are certainly possible. An interesting example is the recent system by Smith and Eisner (2008), which used loopy belief propagation in a dependency parser using highly complex features, while still maintaining cubic-time search complexity.

An obvious drawback of our approach compared to traditional pipeline-based semantic role labeling methods is that the speed of the algorithm is highly dependent on the size of the interdependency feature representation $\boldsymbol{\Phi}_i$. Also, extracting these features is fairly complex, and it is of critical importance to implement the feature extraction procedure efficiently since it is one of the bottlenecks of the algorithm. It is plausible that our performance suffers from the absence of other frequently used syntax-based features such as dependent-of-dependent and voice.

It is thus highly dubious that a joint modeling of syntactic and semantic structure is worth the additional implementational effort. So far, no system using tightly integrated syntactic and semantic processing has been competitive with the best systems, which have been either completely pipeline-based (Che et al., 2008; Ciaramita et al., 2008) or employed only a loose syntactic–semantic coupling (Johansson and Nugues, 2008). It has been conjectured that modeling the semantics of the sentence would also help in syntactic disambiguation; however, it is likely that this is already implicitly taken into account by the lexical features present in virtually all modern parsers.

In addition, a problem that our beam search method has in common with the constituent parsing method by Huang (2008) is that highly non-local features must be computed late. In our case, this means that if there is a long distance between a predicate and an argument, the secondary link between them will be unlikely to influence the final search result.

Acknowledgements

The author is grateful for the helpful comments by the reviewers. This work has been funded by the LivingKnowledge project under the seventh EU framework program.

References

Matthias Buch-Kromann. 2006. *Discontinuous Grammar. A dependency-based model of human parsing and language learning*. Ph.D. thesis, Copenhagen Business School.

Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu, and Sheng Li. 2008. A cascaded syntactic and semantic dependency parsing system. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Massimiliano Ciaramita, Giuseppe Attardi, Felice Dell'Orletta, and Mihai Surdeanu. 2008. DeSRL: A linear-time semantic role labeling system. In *Proceedings of the Shared Task Session of CoNLL-2008*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Schwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 2006(7):551–585.

Ralph Debusmann, Denys Duchier, Alexander Koller, Marco Kuhlmann, Gert Smolka, and Stefan Thater.

2004. A relational syntax-semantics interface based on dependency grammar. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 340–345.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

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

Kevin Gimpel and Noah A. Smith. 2009. Cube summing, approximate inference with non-local features, and dynamic programming without semirings. In *Proceedings of the Twelfth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Jan Hajič. 1998. Building a syntactically annotated corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning*, pages 106–132.

James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT 2005)*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *Proceedings of the Shared Task Session of CoNLL-2008*.

Alex Kulesza and Fernando Pereira. 2008. Structured learning with approximate inference. In *Advances in Neural Information Processing Systems 20*.

Xavier Lluís and Lluís Màrquez. 2008. A joint model for parsing syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98.

Igor A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University Press of New York.

Paola Merlo and Gabriele Musillo. 2008. Semantic parsing for high-precision semantic role labelling. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL–2008)*.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

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

Kenji Sagae and Jun'ichi Tsuji. 2008. Shift–reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Dordrecht:Reidel Publishing Company and Prague:Academia.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Honolulu, United States.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL–2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL–2008*.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Klincksieck, Paris.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of the International Joint Conferences on Artificial Intelligence*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 589–596.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 88–94.

Improving Dependency Parsing with Subtrees from Auto-Parsed Data

Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa
Language Infrastructure Group, MASTAR Project
National Institute of Information and Communications Technology
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan, 619-0289
{chenwl, kazama, uchimoto, torisawa}@nict.go.jp

Abstract

This paper presents a simple and effective approach to improve dependency parsing by using subtrees from auto-parsed data. First, we use a baseline parser to parse large-scale unannotated data. Then we extract subtrees from dependency parse trees in the auto-parsed data. Finally, we construct new subtree-based features for parsing algorithms. To demonstrate the effectiveness of our proposed approach, we present the experimental results on the English Penn Treebank and the Chinese Penn Treebank. These results show that our approach significantly outperforms baseline systems. And, it achieves the best accuracy for the Chinese data and an accuracy which is competitive with the best known systems for the English data.

1 Introduction

Dependency parsing, which attempts to build dependency links between words in a sentence, has experienced a surge of interest in recent times, owing to its usefulness in such applications as machine translation (Nakazawa et al., 2006) and question answering (Cui et al., 2005). To obtain dependency parsers with high accuracy, supervised techniques require a large amount of hand-annotated data. While hand-annotated data are very expensive, large-scale unannotated data can be obtained easily. Therefore, the use of large-scale unannotated data in training is an attractive idea to improve dependency parsing performance.

In this paper, we present an approach that extracts subtrees from dependency trees in auto-parsed data to improve dependency parsing. The auto-parsed data are generated from large-scale unannotated data by using a baseline parser. Then, from dependency trees in the data, we extract different types of subtrees. Finally, we represent subtree-based features on training data to train dependency parsers.

The use of auto-parsed data is not new. However, unlike most of the previous studies (Sagae and Tsujii, 2007; Steedman et al., 2003) that improved the performance by using entire trees from auto-parsed data, we exploit partial information (i.e., subtrees) in auto-parsed data. In their approaches, they used entire auto-parsed trees as newly labeled data to train the parsing models, while we use subtree-based features and employ the original gold-standard data to train the models. The use of subtrees instead of complete trees can be justified by the fact that the accuracy of partial dependencies is much higher than that of entire dependency trees. Previous studies (McDonald and Pereira, 2006; Yamada and Matsumoto, 2003; Zhang and Clark, 2008) show that the accuracies of complete trees are about 40% for English and about 35% for Chinese, while the accuracies of relations between two words are much higher: about 90% for English and about 85% for Chinese. From these observations, we may conjecture that it is possible to conduct a more effective selection by using subtrees as the unit of information.

The use of word pairs in auto-parsed data was tried in van Noord (2007) and Chen et al. (2008). However, the information on word pairs is limited. To provide richer information, we consider more words besides word pairs. Specifically, we use subtrees containing two or three words extracted from dependency trees in the auto-parsed data. To demonstrate the effectiveness of our proposed approach, we present experimental results on En-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 570–579,
Singapore, 6-7 August 2009.

glish and Chinese data. We show that this simple approach greatly improves the accuracy and that the use of richer structures (i.e, word triples) indeed gives additional improvement. We also demonstrate that our approach and other improvement techniques (Koo et al., 2008; Nivre and McDonald, 2008) are complementary and that we can achieve very high accuracies when we combine our method with other improvement techniques. Specifically, we achieve the best accuracy for the Chinese data.

The rest of this paper is as follows: Section 2 introduces the background of dependency parsing. Section 3 proposes an approach for extracting subtrees and represents the subtree-based features for dependency parsers. Section 4 explains the experimental results and Section 5 discusses related work. Finally, in section 6 we draw conclusions.

2 Dependency parsing

Dependency parsing assigns head-dependent relations between the words in a sentence. A simple example is shown in Figure 1, where an arc between two words indicates a dependency relation between them. For example, the arc between "ate" and "fish" indicates that "ate" is the head of "fish" and "fish" is the dependent. The arc between "ROOT" and "ate" indicates that "ate" is the ROOT of the sentence.

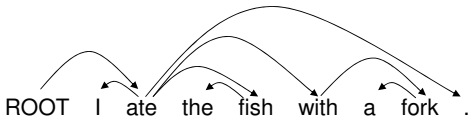


Figure 1: Example for dependency structure

2.1 Parsing approach

For dependency parsing, there are two main types of parsing models (Nivre and McDonald, 2008): graph-based model and transition-based model, which achieved state-of-the-art accuracy for a wide range of languages as shown in recent CoNLL shared tasks (Buchholz et al., 2006; Nivre et al., 2007). Our subtree-based features can be applied in both of the two parsing models.

In this paper, as the base parsing system, we employ the graph-based MST parsing model proposed by McDonald et al. (2005) and McDonald and Pereira (2006), which uses the idea of Maximum Spanning Trees of a graph and large margin structured learning algorithms. The details of parsing model were presented in McDonald et al. (2005) and McDonald and Pereira (2006).

2.2 Baseline Parser

In the MST parsing model, there are two well-used modes: the first-order and the second-order. The first-order model uses first-order features that are defined over single graph edges and the second-order model adds second-order features that are defined on adjacent edges.

For the parsing of unannotated data, we use the first-order MST parsing model, because we need to parse a large number of sentences and the parser must be fast. We call this parser the Baseline Parser.

3 Our approach

In this section, we describe our approach of extracting subtrees from unannotated data. First, we preprocess unannotated data using the Baseline Parser and obtain auto-parsed data. Subsequently, we extract the subtrees from dependency trees in the auto-parsed data. Finally, we generate subtree-based features for the parsing models.

3.1 Subtrees extraction

To ease explanation, we transform the dependency structure into a more tree-like structure as shown in Figure 2, the sentence is the same as the one in Figure 1.

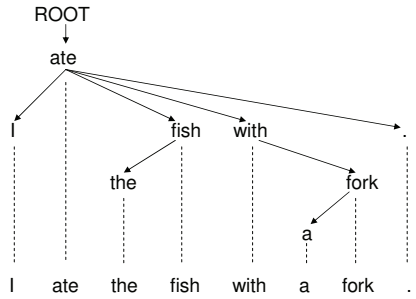


Figure 2: Example for dependency structure in tree-format

Our task is to extract subtrees from dependency trees. If a subtree contains two nodes, we call it a bigram-subtree. If a subtree contains three nodes, we call it a trigram-subtree.

3.2 List of subtrees

We extract subtrees from dependency trees and store them in list L_{st}. First, we extract bigram-subtrees that contain two words. If two words have

a dependency relation in a tree, we add these two words as a subtree into list L_{st}. Similarly, we can extract trigram-subtrees. Note that the dependency direction and the order of the words in the original sentence are important in the extraction. To enable this, the subtrees are encoded in the string format that is expressed as $st = w : wid : hid(-w : wid : hid)+$[1], where w refers to a word in the subtree, wid refers to the ID (starting from 1) of a word in the subtree (words are ordered according to the positions of the original sentence)[2] , and hid refers to an ID of the head of the word (hid=0 means that this word is the root of a subtree). For example, "ate" and "fish" have a right dependency arc in the sentence shown in Figure 2. So the subtree is encoded as "ate:1:0-fish:2:1". Figure 3 shows all the subtrees extracted from the sentence in Figure 2, where the subtrees in (a) are bigram-subtrees and the ones in (b) are trigram-subtrees.

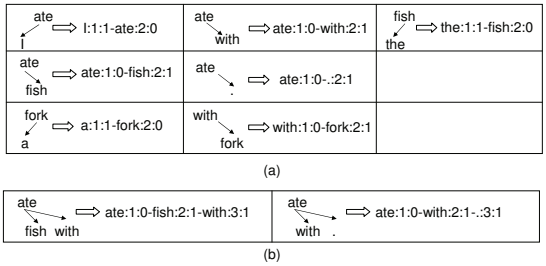


Figure 3: Examples of subtrees

Note that we only used the trigram-subtrees containing a head, its dependent $d1$, and $d1$'s leftmost right sibling[3]. We could not consider the case where two children are on different sides[4] of the head (for instance, "I" and "fish" for "ate" in Figure 2). We also do not use the child-parent-grandparent type (grandparent-type in short) trigram-subtrees. These are due to the limitations of the parsing algorithm of (McDonald and Pereira, 2006), which does not allow the features defined on those types of trigram-subtrees.

We extract the subtrees from the auto-parsed data, then merge the same subtrees into one entry, and count their frequency. We eliminate all subtrees that occur only once in the data.

[1] + refers to matching the preceding element one or more times and is the same as a regular expression in Perl.

[2] So, wid is in fact redundant but we include it for ease of understanding.

[3] Note that the order of the siblings is based on the order of the words in the original sentence.

[4] Here, "side" means the position of a word relative to the head in the original sentence.

3.3 Subtree-based features

We represent new features based on the extracted subtrees and call them subtree-based features. The features based on bigram-subtrees correspond to the first-order features in the MST parsing model and those based on trigram-subtrees features correspond to the second-order features.

We first group the extracted subtrees into different sets based on their frequencies. After experiments with many different threshold settings on development data sets, we chose the following way. We group the subtrees into three sets corresponding to three levels of frequency: "high-frequency (HF)", "middle-frequency (MF)", and "low-frequency (LF)". HF, MF, and LF are used as set IDs for the three sets. The following are the settings: if a subtree is one of the TOP-10% most frequent subtrees, it is in set HF; else if a subtree is one of the TOP-20% subtrees, it is in set MF; else it is in set LF. Note that we compute these levels within a set of subtrees with the same number of nodes. We store the set ID for every subtree in L_{st}. For example, if subtree "ate:1:0-with:2:1" is among the TOP-10%, its set ID is HF.

3.3.1 First-order subtree-based features

The first-order features are based on bigram-subtrees that are related to word pairs. We generate new features for a head h and a dependent d in the parsing process. Figure 4-(a)[5] shows the words and their surrounding words, where h_{-1} refers to the word to the left of the head in the sentence, h_{+1} refers to the word to the right of the head, d_{-1} refers to the word to the left of the dependent, and d_{+1} refers to the word to the right of the dependent. Temporary bigram-subtrees are formed by word pairs that are linked by dashed-lines in the figure. Then we retrieve these subtrees in L_{st} to get their set IDs (if a subtree is not included in L_{st}, its set ID is ZERO. That is, we have four sets: HF, MF, LF, and ZERO.).

Then we generate first-order subtree-based features, consisting of indicator functions for set IDs of the retrieved bigram-subtrees. When generating subtree-based features, each dashed line in Figure 4-(a) triggers a different feature.

To demonstrate how to generate first-order subtree-based features, we use an example that is as follows. Suppose that we are going to parse the sentence "He ate the cake with a fork." as shown

[5] Please note that d could be before h.

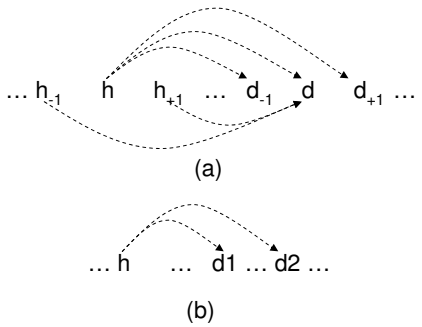


Figure 4: Word pairs and triple for feature representation

in Figure 5, where h is "ate" and d is "with". We can generate the features for the pairs linked by dashed-lines, such as $h - d$, $h - d_{+1}$ and so on. Then we have the temporary bigram-subtrees "ate:1:0-with:2:1" for $h - d$ and "ate:1:0-a:2:1" for $h - d_{+1}$, and so on. If we can find subtree "ate:1:0-with:2:1" for $h - d$ from L_{st} with set ID HF, we generate the feature "H-D:HF", and if we find subtree "ate:1:0-a:2:1" for $h - d_{+1}$ with set ID ZERO, we generate the feature "H-D+1:ZERO". The other three features are also generated similarly.

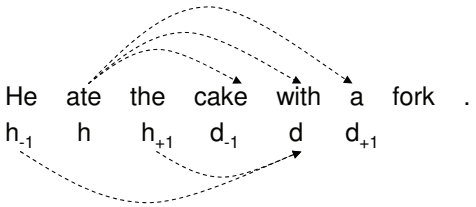


Figure 5: First-order subtree-based features

3.3.2 Second-order subtree-based features

The second-order features are based on trigram-subtrees that are related to triples of words. We generate features for a triple of a head h, its dependent $d1$, and $d1$'s right-leftmost sibling $d2$. The triple is shown in Figure 4-(b). A temporary trigram-subtree is formed by the word forms of h, $d1$, and $d2$. Then we retrieve the subtree in L_{st} to get its set ID. In addition, we consider the triples of "h-NULL"[6], $d1$, and $d2$, which means that we only check the words of sibling nodes without checking the head word.

Then, we generate second-order subtree-based features, consisting of indicator functions for set IDs of the retrieved trigram-subtrees.

[6]h-NULL is a dummy token

We also generate combined features involving the set IDs and part-of-speech tags of heads, and the set IDs and word forms of heads. Specifically, for any feature related to word form, we remove this feature if the word is not one of the Top-N most frequent words in the training data. We used N=1000 for the experiments in this paper. This method can reduce the size of the feature sets.

In this paper, we only used bigram-subtrees and the limited form of trigram-subtrees, though in theory we can use k-gram-subtrees, which are limited in the same way as our trigram subtrees, in (k-1)th-order MST parsing models mentioned in McDonald and Pereira (2006) or use grandparent-type trigram-subtrees in parsing models of Carreras (2007). Although the higher-order MST parsing models will be slow with exact inference, requiring O(n^k) time (McDonald and Pereira, 2006), it might be possible to use higher-order k-gram subtrees with approximated parsing model in the future. Of course, our method can also be easily extended to the labeled dependency case.

4 Experiments

In order to evaluate the effectiveness of the subtree-based features, we conducted experiments on English data and Chinese Data.

For English, we used the Penn Treebank (Marcus et al., 1993) in our experiments and the tool "Penn2Malt"[7] to convert the data into dependency structures using a standard set of head rules (Yamada and Matsumoto, 2003). To match previous work (McDonald et al., 2005; McDonald and Pereira, 2006; Koo et al., 2008), we split the data into a training set (sections 2-21), a development set (Section 22), and a test set (section 23). Following the work of Koo et al. (2008), we used the MXPOST (Ratnaparkhi, 1996) tagger trained on training data to provide part-of-speech tags for the development and the test set, and we used 10-way jackknifing to generate tags for the training set. For the unannotated data, we used the BLLIP corpus (Charniak et al., 2000) that contains about 43 million words of WSJ text.[8] We used the MXPOST tagger trained on training data to assign part-of-speech tags and used the Basic Parser to process the sentences of the BLLIP corpus.

For Chinese, we used the Chinese Treebank

[7]http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html

[8]We ensured that the text used for extracting subtrees did not include the sentences of the Penn Treebank.

(CTB) version 4.0[9] in the experiments. We also used the "Penn2Malt" tool to convert the data and created a data split: files 1-270 and files 400-931 for training, files 271-300 for testing, and files 301-325 for development. We used gold standard segmentation and part-of-speech tags in the CTB. The data partition and part-of-speech settings were chosen to match previous work (Chen et al., 2008; Yu et al., 2008). For the unannotated data, we used the PFR corpus[10], which has approximately 15 million words whose segmentation and POS tags are given. We used its original segmentation though there are differences in segmentation policy between CTB and this corpus. As for POS tags, we discarded the original POS tags and assigned CTB style POS tags using a TNT-based tagger (Brants, 2000) trained on the training data. We used the Basic Parser to process all the sentences of the PFR corpus.

We measured the parser quality by the unlabeled attachment score (UAS), i.e., the percentage of tokens (excluding all punctuation tokens) with the correct HEAD. And we also evaluated on complete dependency analysis.

4.1 Experimental Results

In our experiments, we used MSTParser, a freely available implementation[11] of the first- and second-order MST parsing models. For baseline systems, we used the first- and second-order basic features, which were the same as the features used by McDonald and Pereira (2006), and we used the default settings of MSTParser throughout the paper: iters=10; training-k=1; decode-type=proj. We implemented our systems based on the MSTParser by incorporating the subtree-based features.

4.1.1 Main results of English data

	English	
	UAS	Complete
Ord1	90.95	37.45
Ord1s	91.76(+0.81)	40.68
Ord2	91.71	42.88
Ord2s	92.51(+0.80)	46.19
Ord2b	92.28(+0.57)	45.44
Ord2t	92.06(+0.35)	42.96

Table 1: Dependency parsing results for English

[9]http://www.cis.upenn.edu/~chinese/.

[10]http://www.icl.pku.edu.

[11]http://mstparser.sourceforge.net

The results are shown in Table 1, where Ord1/Ord2 refers to a first-/second-order MSTParser with basic features, Ord1s/Ord2s refers to a first-/second-order MSTParser with basic+subtree-based features, and the improvements by the subtree-based features over the basic features are shown in parentheses. Note that we use both the bigram- and trigram- subtrees in Ord2s. The parsers using the subtree-based features consistently outperformed those using the basic features. For the first-order parser, we found that there is an absolute improvement of 0.81 points (UAS) by adding subtree-based features. For the second-order parser, we got an absolute improvement of 0.8 points (UAS) by including subtree-based features. The improvements of parsing with subtree-based features were significant in McNemar's Test ($p < 10^{-6}$).

We also checked the sole effect of bigram- and trigram-subtrees. The results are also shown in Table 1, where Ord2b/Ord2t refers to a second-order MSTParser with bigram-/trigram-subtrees only. The results showed that trigram-subtrees can provide further improvement, although the effect of the bigram-subtrees seemed larger.

4.1.2 Comparative results of English data

Table 2 shows the performance of the systems that were compared, where Y&M2003 refers to the parser of Yamada and Matsumoto (2003), CO2006 refers to the parser of Corston-Oliver et al. (2006), Hall2006 refers to the parser of Hall et al. (2006), Wang2007 refers to the parser of Wang et al. (2007), Z&C 2008 refers to the combination graph-based and transition-based system of Zhang and Clark (2008), KOO08-dep1c/KOO08-dep2c refers to a graph-based system with first-/second-order cluster-based features by Koo et al. (2008), and Carreras2008 refers to the paper of Carreras et al. (2008). The results showed that Ord2s performed better than the first five systems. The second-order system of Koo et al. (2008) performed better than our systems. The reason may be that the MSTParser only uses sibling interactions for second-order, while Koo et al. (2008) uses both sibling and grandparent interactions, and uses cluster-based features. Carreras et al. (2008) reported a very high accuracy using information of constituent structure of the TAG grammar formalism. In our systems, we did not use such knowledge.

Our subtree-based features could be combined

with the techniques presented in other work, such as the cluster-based features in Koo et al. (2008), the integrating methods of Zhang and Clark (2008), and Nivre and McDonald (2008), and the parsing methods of Carreras et al. (2008).

	English	
	UAS	Complete
Y&M2003	90.3	38.4
CO2006	90.8	37.6
Hall2006	89.4	36.4
Wang2007	89.2	34.4
Z&C2008	92.1	45.4
KOO08-dep1c	92.23	–
KOO08-dep2c	93.16	–
Carreras2008	93.5	–
Ord1	90.95	37.45
Ord1s	91.76	40.68
Ord1c	91.88	40.71
Ord1i	91.68	41.43
Ord1sc	92.20	42.98
Ord1sci	92.60	44.28
Ord2	91.71	42.88
Ord2s	92.51	46.19
Ord2c	92.40	44.08
Ord2i	92.12	44.37
Ord2sc	92.70	46.56
Ord2sci	93.16	47.15

Table 2: Dependency parsing results for English, for our parsers and previous work

To demonstrate that our approach and other work are complementary, we thus implemented a system using all the techniques we had at hand that used subtree- and cluster-based features and applied the integrating method of Nivre and McDonald (2008). We used the word clustering tool[12], which was used by Koo et al. (2008), to produce word clusters on the BLLIP corpus. The cluster-based features were the same as the features used by Koo et al. (2008). For the integrating method, we used the transition MaxEnt-based parser of Zhao and Kit (2008) because it was faster than the MaltParser. The results are shown in the bottom part of Table 2, where Ord1c/Ord2c refers to a first-/second-order MSTParser with cluster-based features, Ord1i/Ordli refers to a first-/second-order MSTParser with integrating-based features, Ord1sc/Ord2sc refers to a first-/second-order MSTParser with subtree-based+cluster-based features, and Ord1sci/Ord2sci refers to a first-/second-order MSTParser with subtree-based+cluster-based+integrating-based features. Ord1c/Ord2c was worse than KOO08-dep1c/-dep2c, but Ord1sci outperformed KOO08-dep1c and Ord2sci performed similarly to KOO08-dep2c by using all of the techniques we had. These results indicated that subtree-based features can provide different information and work well with other techniques.

[12]http://www.cs.berkeley.edu/~pliang/software/brown-cluster-1.2.zip

4.1.3 Main results of Chinese data

The results are shown in Table 3 where abbreviations are the same as in Table 1. As in the English experiments, parsers with the subtree-based features outperformed parsers with the basic features, and second-order parsers outperformed first-order parsers. For the first-order parser, the subtree-based features provided 1.3 absolute points improvement. For the second-order parser, the subtree-based features achieved an absolute improvement of 1.25 points. The improvements of parsing with subtree-based features were significant in McNemar's Test ($p < 10^{-5}$).

	Chinese	
	UAS	Complete
Ord1	86.38	40.80
Ord1s	87.68(+1.30)	42.24
Ord2	88.18	47.12
Ord2s	89.43(+1.25)	47.53
Ord2b	89.16(+0.98)	47.12
Ord2t	88.55(+0.37)	47.12

Table 3: Dependency parsing results for Chinese.

4.1.4 Comparative results of Chinese data

Table 4 shows the comparative results, where Wang2007 refers to the parser of Wang et al. (2007), Chen2008 refers to the parser of Chen et al. (2008), and Yu2008 refers to the parser of Yu et al. (2008) that is the best reported results for this data set. And "all words" refers to all the sentences in test set and "$\leq$ 40 words"[13] refers to the sentences with the length up to 40. The table shows that our parsers outperformed previous systems.

We also implemented integrating systems for Chinese data as well. When we applied the cluster-based features, the performance dropped a little. The reason may be that we are using gold-POS tags for Chinese data[14]. Thus we did not

[13]Wang et al. (2007) and Chen et al. (2008) reported the scores on these sentences.

[14]We tried to use the cluster-based features for Chinese with the same setting of POS tags as English data, then the cluster-based features did provide improvement.

use cluster-based features for the integrating systems. The results are shown in Table 4, where Ord1si/Ord2si refers to the first-order/second-order system with subtree-based+intergrating-based features. We found that the integrating systems provided better results. Overall, we have achieved a high accuracy, which is the best known result for this dataset.

Zhang and Clark (2008) and Duan et al. (2007) reported results on a different data split of Penn Chinese Treebank. We also ran our systems (Ord2s) on their data and provided UAS 86.70 (for non-root words)/77.39 (for root words), better than their results: 86.21/76.26 in Zhang and Clark (2008) and 84.36/73.70 in Duan et al. (2007).

	Chinese			
	all words		$\leq$ 40 words	
	UAS	Complete	UAS	Complete
Wang2007	–	–	86.6	28.4
Chen2008	86.52	–	88.4	–
Yu2008	87.26	–	–	–
Ord1s	87.68	42.24	91.11	54.40
Ord1si	88.24	43.96	91.32	55.93
Ord2s	89.43	47.53	91.67	59.77
Ord2si	89.91	48.56	92.34	62.83

Table 4: Dependency parsing results for Chinese, for our parsers and for previous work

4.1.5 Effect of different sizes of unannotated data

Here, we consider the improvement relative to the sizes of the unannotated data. Figure 6 shows the results of first-order parsers with different numbers of words in the unannotated data. Please note that the size of full English unannotated data is 43M and the size of full Chinese unannotated data is 15M. From the figure, we found that the parser obtained more benefits as we added more unannotated data.

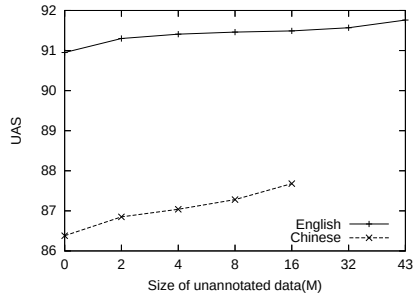


Figure 6: Results with different sizes of large-scale unannotated data.

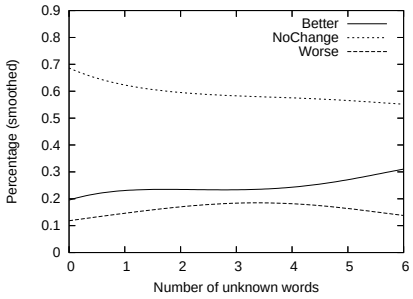


Figure 7: Improvement relative to unknown words for English

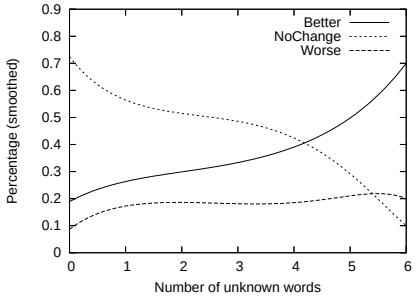


Figure 8: Improvement relative to unknown words for Chinese

4.2 Additional Analysis

In this section, we investigated the results on sentence level from different views. For Figures 7-12, we classified each sentence into one of three classes: "Better" for those where the proposed parsers provided better results relative to the parsers with basic features, "Worse" for those where the proposed parsers provided worse results relative to the basic parsers, and "NoChange" for those where the accuracies remained the same.

4.2.1 Unknown words

Here, we consider the unknown word[15] problem, which is an important issue for parsing. We calculated the number of unknown words in one sentence, and listed the changes of the sentences with unknown words. Here, we compared the Ord1 system and the Ord1s system.

Figures 7 and 8 show the results, where the x axis refers to the number of unknown words in one sentence and the y axis shows the percentages of the three classes. For example, for the sentences having three unknown words in the Chinese data, 31.58% improved, 23.68% worsened, and 44.74% were unchanged. We did not show the results of

[15] An unknown word is a word that is not included in the training data.

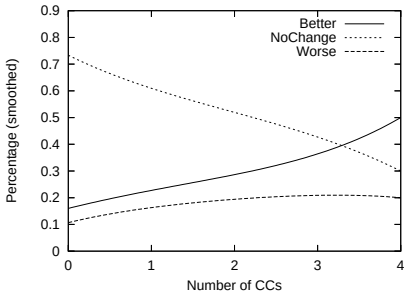


Figure 9: Improvement relative to number of conjunctions for English

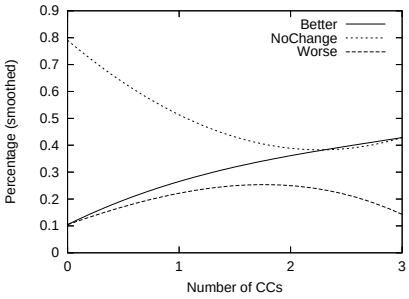


Figure 10: Improvement relative to number of conjunctions for Chinese

the sentences with more than six unknown words because their numbers were very small. The Better and Worse curves showed that our approach always provided better results. The results indicated that the improvements apparently became larger when the sentences had more unknown words for the Chinese data. And for the English data, the graph also showed the similar trend, although the improvements for the sentences have three and four unknown words were slightly less than the others.

4.2.2 Coordinating conjunctions

We analyzed our new parsers' behavior for coordinating conjunction structures, which is a very difficult problem for parsing (Kawahara and Kurohashi, 2008). Here, we compared the Ord2 system with the Ord2s system.

Figures 9 and 10 show how the subtree-based features affect accuracy as a function of the number of conjunctions, where the x axis refers to the number of conjunctions in one sentence and the y axis shows the percentages of the three classes. The figures indicated that the subtree-based features improved the coordinating conjunction problem. In the trigram-subtree list, many subtrees are related to coordinating conjunctions, such as "utilities:1:3 and:2:3 businesses:3:0" and "pull:1:0 and:2:1 protect:3:1". These subtrees can provide additional information for parsing models.

4.2.3 PP attachment

We analyzed our new parsers' behavior for preposition-phrase attachment, which is also a difficult task for parsing (Ratnaparkhi et al., 1994). We compared the Ord2 system with the Ord2s system. Figures 11 and 12 show how the subtree-based features affect accuracy as a function of the number of prepositions, where the x axis refers to the number of prepositions in one sentence and the y axis shows the percentages of the three classes. The figures indicated that the subtree-based features improved preposition-phrase attachment.

5 Related work

Our approach is to incorporate unannotated data into parsing models for dependency parsing. Several previous studies relevant to our approach have been conducted.

Chen et al. (2008) previously proposed an approach that used the information on short dependency relations for Chinese dependency parsing. They only used the word pairs within two word distances for a transition-based parsing algorithm. The approach in this paper differs in that we use richer information on trigram-subtrees besides bigram-subtrees that contain word pairs. And our work is focused on graph-based parsing models as opposed to transition-based models. Yu et al. (2008) constructed case structures from auto-parsed data and utilized them in parsing. Compared with their method, our method is much simpler but has great effects.

Koo et al. (2008) used the Brown algorithm to produce word clusters on large-scale unannotated data and represented new features based on the clusters for parsing models. The cluster-based features provided very impressive results. In addition, they used the parsing model by Carreras (2007) that applied second-order features on both sibling and grandparent interactions. Note that our approach and their approach are complementary in that we can use both subtree- and cluster-based features for parsing models. The experimental results showed that we achieved better accuracy for first-order models when we used both of these two types of features.

Sagae and Tsujii (2007) presented an co-training approach for dependency parsing adap-

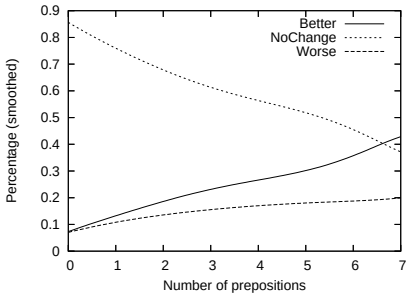


Figure 11: Improvement relative to number of prepositions for English

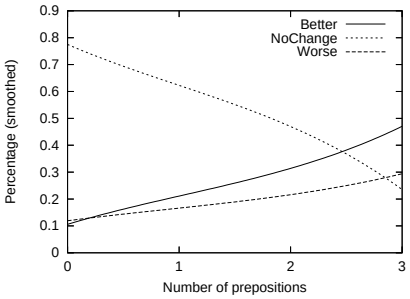


Figure 12: Improvement relative to number of prepositions for Chinese

tation. They used two parsers to parse the sentences in unannotated data and selected only identical results produced by the two parsers. Then, they retrained a parser on newly parsed sentences and the original labeled data. Our approach represents subtree-based features on the original gold-standard data to retrain parsers. McClosky et al. (2006) presented a self-training approach for phrase structure parsing and the approach was shown to be effective in practice. However, their approach depends on a high-quality reranker, while we simply augment the features of an existing parser. Moreover, we could use the output of our systems for co-training/self-training techniques.

6 Conclusions

We present a simple and effective approach to improve dependency parsing using subtrees from auto-parsed data. In our method, first we use a baseline parser to parse large-scale unannotated data, and then we extract subtrees from dependency parsing trees in the auto-parsed data. Finally, we construct new subtree-based features for parsing models. The results show that our approach significantly outperforms baseline systems. We also show that our approach and other techniques are complementary, and then achieve the best reported accuracy for the Chinese data and an accuracy that is competitive with the best known systems for the English data.

References

T. Brants. 2000. TnT–a statistical part-of-speech tagger. *Proceedings of ANLP*, pages 224–231.

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X shared task on multilingual dependency parsing. *Proceedings of CoNLL-X*.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of CoNLL 2008*, pages 9–16, Manchester, England, August. Coling 2008 Organizing Committee.

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961.

E. Charniak, D. Blaheta, N. Ge, K. Hall, J. Hale, and M. Johnson. 2000. BLLIP 1987-89 WSJ Corpus Release 1, LDC2000T43. *Linguistic Data Consortium*.

WL. Chen, D. Kawahara, K. Uchimoto, YJ. Zhang, and H. Isahara. 2008. Dependency parsing with short dependency relations in unlabeled data. In *Proceedings of IJCNLP 2008*.

S. Corston-Oliver, A. Aue, Kevin. Duh, and Eric Ringger. 2006. Multilingual dependency parsing using bayes point machines. In *HLT-NAACL2006*.

H. Cui, RX. Sun, KY. Li, MY. Kan, and TS. Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of SIGIR 2005*, pages 400–407, New York, NY, USA. ACM.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic models for action-based chinese dependency parsing. In *Proceedings of ECML/ECPPKDD*, Warsaw, Poland.

Johan Hall, Joakim Nivre, and Jens Nilsson. 2006. Discriminative classifiers for deterministic dependency parsing. In *In Proceedings of CoLING-ACL*.

D. Kawahara and S. Kurohashi. 2008. Coordination disambiguation without any similarities. In *Proceedings of Coling 2008*, pages 425–432, Manchester, UK, August.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, June.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguisticss*, 19(2):313–330.

D. McClosky, E. Charniak, and M. Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of Coling-ACL*, pages 337–344.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL2006*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL 2005*.

T. Nakazawa, K. Yu, D. Kawahara, and S. Kurohashi. 2006. Example-based machine translation based on deeper nlp. In *Proceedings of IWSLT 2006*, pages 64–70, Kyoto, Japan.

J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, June.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

A. Ratnaparkhi, J. Reynar, and S. Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proceedings of HLT*, pages 250–255.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*, pages 133–142.

K. Sagae and J. Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050.

M. Steedman, M. Osborne, A. Sarkar, S. Clark, R. Hwa, J. Hockenmaier, P. Ruhlen, S. Baker, and J. Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of EACL 2003*, pages 331–338.

Gertjan van Noord. 2007. Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of IWPT-07*, June.

Qin Iris Wang, Dekang Lin, and Dale Schuurmans. 2007. Simple training of dependency parsers via structured boosting. In *Proceedings of IJCAI2007*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT2003*, pages 195–206.

K. Yu, D. Kawahara, and S. Kurohashi. 2008. Chinese dependency parsing with large scale automatically constructed case structures. In *Proceedings of Coling 2008*, pages 1049–1056, Manchester, UK, August.

Y. Zhang and S. Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of EMNLP 2008*, pages 562–571, Honolulu, Hawaii, October.

H. Zhao and CY. Kit. 2008. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *Proceedings of CoNLL 2008*, pages 203–207, Manchester, England, August.

Topic-wise, Sentiment-wise, or Otherwise? Identifying the Hidden Dimension for Unsupervised Text Classification

Sajib Dasgupta and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{sajib,vince}@hlt.utdallas.edu

Abstract

While traditional work on text clustering has largely focused on grouping documents by topic, it is conceivable that a user may want to cluster documents along other dimensions, such as the author's mood, gender, age, or sentiment. Without knowing the user's intention, a clustering algorithm will only group documents along the most prominent dimension, which may not be the one the user desires. To address this problem, we propose a novel way of incorporating user feedback into a clustering algorithm, which allows a user to easily specify the dimension along which she wants the data points to be clustered via inspecting only a small number of words. This distinguishes our method from existing ones, which typically require a large amount of effort on the part of humans in the form of document annotation or interactive construction of the feature space. We demonstrate the viability of our method on several challenging sentiment datasets.

1 Introduction

Text clustering is one of the most important applications in Natural Language Processing (NLP). A common approach to this problem consists of (1) computing the similarity between each pair of documents, each of which is typically represented as a bag of words; and (2) using an unsupervised clustering algorithm to partition the documents. The majority of existing work on text clustering has focused on *topic-based* clustering, where high accuracies can be achieved even for datasets with a large number of classes (e.g., 20 Newsgroups).

On the other hand, there has been relatively little work on *sentiment-based* clustering and the related task of *unsupervised polarity classification*, where the goal is to cluster (or classify) a set of documents (e.g., reviews) according to the polarity (e.g., "thumbs up" or "thumbs down") expressed by the author in an unsupervised manner. Despite the large amount of recent work on sentiment analysis and opinion mining, much of it has focused on *supervised* methods (e.g., Pang et al. (2002), Kim and Hovy (2004), Mullen and Collier (2004)). One weakness of these existing supervised polarity classification systems is that they are typically *domain-* and *language-specific*. Hence, when given a new domain or language, one needs to go through the expensive process of collecting a large amount of annotated data in order to train a high-performance polarity classifier. Some recent attempts have been made to leverage existing sentiment corpora or lexica to automatically create annotated resources for new domains or languages. However, such methods require the existence of either a parallel corpus/machine translation engine for projecting/translating annotations/lexica from a resource-rich language to the target language (Banea et al., 2008; Wan, 2008), or a domain that is "similar" enough to the target domain (Blitzer et al., 2007). When the target domain or language fails to meet this requirement, sentiment-based clustering or unsupervised polarity classification become appealing alternatives. Unfortunately, to our knowledge, these tasks are largely under-investigated in the NLP community. Turney's (2002) work is perhaps one of the most notable examples of unsupervised polarity classification. However, while his system learns the semantic orientation of the phrases in a review in an unsupervised manner, this information is used to predict the polarity of a review heuristically.

Despite its practical significance, sentiment-based clustering is a challenging task. To illustrate its difficulty, consider the task of clustering a set of movie reviews. Since each review may contain a description of the plot and the author's

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 580–589,
Singapore, 6-7 August 2009.

sentiment, a clustering algorithm may cluster reviews along either the *plot* dimension or the *sentiment* dimension; and without knowing the user's intention, they will be clustered along the most *prominent* dimension. Assuming the usual bag-of-words representation, the most prominent dimension will more likely be *plot*, as it is not uncommon for a review to be devoted almost exclusively to the plot, with the author briefly expressing her sentiment only at the end of the review. Even if the reviews contain mostly subjective material, the most prominent dimension may still not be *sentiment*, due to the fact that many reviews are *sentimentally ambiguous*. Specifically, a reviewer may have negative opinions on the actors but at the same time talk enthusiastically about how much she enjoyed the plot. The presence of both positive and negative sentiment-bearing words in these reviews renders the sentiment dimension *hidden* (i.e., less prominent) as far as clustering is concerned. Therefore, there is no guarantee that the clustering algorithm will automatically produce a sentiment-based clustering of the reviews.

Hence, it is important for a user to provide feedback on the clustering process to ensure that the reviews are clustered along the *sentiment* dimension, possibly in an interactive manner. One way to do this would be to ask the user to annotate a small number of reviews with polarity information, possibly through an active learning procedure to minimize human intervention (Dredze and Crammer, 2008). Another way would be to have the user explicitly identify the relevant features (in our case, the sentiment-bearing words) at the beginning of the clustering process (Liu et al., 2004), or incrementally construct the set of relevant features in an interactive fashion (Bekkerman et al., 2007; Raghavan and Allan, 2007; Roth and Small, 2009). In addition, the user may supply constraints on which pairs of documents must or must not appear in the same cluster (Wagstaff et al., 2001), or simply tell the algorithm whether two clusters should be *merged* or *split* during the clustering process (Balcan and Blum, 2008). It is worth noting that many of these feedback mechanisms were developed by machine learning researchers for general clustering tasks and not for sentiment-based clustering.

Our goal in this paper is to propose a novel mechanism allowing a user to cluster a set of documents along the desired dimension, which may be a hidden dimension, with *very limited* user feedback. In comparison to the aforementioned feedback mechanisms, ours is arguably much simpler: we only require that the user *select* a dimension by examining a small number of features for each dimension, as opposed to having the user *generate* the feature space in an interactive manner or *identify* clusters that need to be merged or split. In particular, identifying clusters for merging or splitting in Balcan and Blum's algorithm may not be as easy as it appears: for each MERGE or SPLIT decision the user makes, she has to sample a large number of documents from the cluster(s), read through the documents, and base her decision on the extent to which the documents are (dis)similar to each other. Perhaps more importantly, our human experiments involving five users indicate that all of them can easily identify the sentiment dimension based on the features, thus providing suggestive evidence that our method is viable.

In sum, our contributions in this paper are threefold. First, we propose a novel feedback mechanism for clustering allowing a user to easily specify the dimension along which she wants data points to be clustered and apply the mechanism to the challenging, yet under-investigated problem of sentiment-based clustering. Second, spectral learning, which is the core of our method, has not been applied extensively to NLP problems, and we hope that our work can increase the awareness of this powerful machine learning technique in the NLP community. Finally, we demonstrate the viability of our method not only by evaluating its performance on sentiment datasets, but also via a set of human experiments, which is typically absent in papers that involve algorithms for incorporating user feedback.

The rest of the paper is organized as follows. Section 2 presents the basics of spectral clustering, which will facilitate the discussion of our feedback mechanism in Section 3. We describe our human experiments and evaluation results on several sentiment datasets in Section 4, and present our conclusions in Section 5.

2 Spectral Clustering

When given a clustering task, an important question to ask is: which clustering algorithm should we use? A popular choice is k-means. Nevertheless, it is well-known that k-means has the major drawback of not being able to separate data points

that are not linearly separable in the given feature space (e.g., see Dhillon et al. (2004) and Cai et al. (2005)). Spectral clustering algorithms were developed in response to this problem with k-means. The central idea behind spectral clustering is to (1) construct a low-dimensional space from the original (typically high-dimensional) space while retaining as much information about the original space as possible, and (2) cluster the data points in this low-dimensional space. The rest of this section provides the details of spectral clustering.

2.1 Algorithm

Although there are several well-known spectral clustering algorithms in the literature (e.g., Weiss (1999), Shi and Malik (2000), Kannan et al. (2004)), we adopt the one proposed by Ng et al. (2002), as it is arguably the most widely-used. The algorithm takes as input a similarity matrix S created by applying a user-defined similarity function to each pair of data points. Below are the main steps of the algorithm:

1. Create the diagonal matrix D whose (i,i)-th entry is the sum of the i-th row of S, and then construct the Laplacian matrix $L = D^{-1/2}SD^{-1/2}$.
2. Find the eigenvalues and eigenvectors of L.
3. Create a new matrix from the m eigenvectors that correspond to the m largest eigenvalues.[1]
4. Each data point is now rank-reduced to a point in the m-dimensional space. Normalize each point to unit length (while retaining the sign of each value).
5. Cluster the resulting data points using k-means.

In essence, each dimension in the reduced space is defined by exactly one eigenvector. The reason why eigenvectors with large eigenvalues are used is that they capture the largest variance in the data. As a result, each of them can be thought of as revealing an important dimension of the data.

2.2 Clustering with Eigenvectors

As Ng et al. (2002) point out, "different authors still disagree on which eigenvectors to use, and how to derive clusters from them". There are two common methods for deriving clusters using the eigenvectors. These methods will serve as our baselines in our evaluation.

[1]For brevity, we will refer to the eigenvector with the n-th largest eigenvalue simply as the n-th eigenvector.

Method 1: Using the second eigenvector only
The first method is to use only the second eigenvector, $\mathbf{e}_2$, to partition the points. Besides revealing one of the most important dimensions of the data, this eigenvector induces an intuitively ideal partition of the data — the partition induced by the minimum normalized cut of the similarity graph[2], where the nodes are the data points and the edge weights are the pairwise similarity values of the points (Shi and Malik, 2000). Clustering in a one-dimensional space is trivial: since we have a linearization of the points, all we need to do is to determine a threshold for partitioning the points. However, we follow Ng et al. (2002) and cluster using 2-means in this one-dimensional space.

Method 2: Using m eigenvectors
Recall from Section 2.1 that after eigen-decomposing the Laplacian matrix, each data point is represented by m co-ordinates. In the second method, we simply use 2-means to cluster the data points in this m-dimensional space, effectively exploiting all of the m eigenvectors.

3 Our Approach

As mentioned before, sentiment-based clustering is challenging, in part due to the fact that the reviews can be clustered along more than one dimension. In this section, we propose and incorporate a user feedback mechanism into a spectral clustering algorithm, which makes it easy for a user to specify the dimension along which she wants to cluster the data points.

Recall that our method first applies spectral clustering to reveal the most important dimensions of the data, and then lets the user select the desired dimension. To motivate the importance of user feedback, it helps to understand why the two baseline clustering algorithms described in Section 2.2, which are also based on spectral methods but do not rely on user feedback, may not always yield a sentiment-based clustering. To begin with, consider the first method, where only the second eigenvector is used to induce the partition. Recall that the second eigenvector reveals the most prominent dimension of the data. Hence, if sentiment is not the most prominent dimension (which can happen if the non-sentiment-bearing

[2]Using the normalized cut (as opposed to the usual cut) ensures that the size of the two clusters are relatively balanced, avoiding trivial cuts where one cluster is empty and the other is full. See Shi and Malik (2000) for details.

words outnumber the sentiment-bearing words in the bag-of-words representation of a review), then the resulting clustering of the reviews may not be sentiment-oriented. A similar line of reasoning can be used to explain why the second baseline clustering algorithm, which clusters based on all of the eigenvectors in the low-dimensional space, may not always work well. Since each eigenvector corresponds to a different dimension (and, in particular, some of them correspond to non-sentiment dimensions), using all of them to represent a review may hamper the accurate computation of the similarity of two reviews as far as clustering along the sentiment dimension is concerned. In the rest of this section, we discuss the major steps of our user-feedback mechanism in detail.

Step 1: Identify the important dimensions
To identify the important dimensions of the given reviews, we take the top eigenvectors computed from the eigen-decomposition of the Laplacian matrix, which is in turn formed from the input similarity matrix. We compute the similarity between two reviews by taking the dot product of their feature vectors (see Section 4.1 for details on feature vector generation). Following Ng et al., we set the diagonal entries of the similarity matrix to 0.

Step 2: Identify the relevant features
Given the eigen-decomposition from Step 1, we first obtain the second through the fifth eigenvectors[3], which as mentioned above, correspond to the most important dimensions of the data. Then, we ask the user to select one of the four dimensions defined by these eigenvectors according to their relevance to sentiment. One way to do this is to (1) induce one partition of the reviews from each of the four eigenvectors, using a procedure identical to Method 1 in Section 2.2, and (2) have the user inspect the four partitions and decide which corresponds most closely to a sentiment-based clustering. The main drawback associated with this kind of user feedback is that the user may have to read a large number of reviews in order to make a decision. Hence, to reduce human effort, we employ an alternative procedure: we (1) identify the most informative features for characterizing each partition, and (2) have the user inspect just the features rather than the reviews.

While traditional feature selection techniques such as log-likelihood ratio and information gain can be applied to identify these informative features (see Yang and Pedersen (1997) for an overview), we employ a more sophisticated feature-ranking method that we call *maximum margin feature ranking* (MMFR). Recall that a maximum margin classifier (e.g., a support vector machine) separates data points from two classes while maximizing the margin of separation. Specifically, a maximum margin hyperplane is defined by $\mathbf{w} \cdot \mathbf{x} - b = 0$, where $\mathbf{x}$ is a feature vector representing an arbitrary data point, and $\mathbf{w}$ (a weight vector) and b (a scalar) are parameters that are learned by solving the following constrained optimization problem:

$$\arg\min \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i$$

subject to

$$c_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1 - \xi_i, \quad 1 \leq i \leq n,$$

where $c_i \in \{+1, -1\}$ is the class of the i-th training point $\mathbf{x}_i$, ξ_i is the degree of misclassification of $\mathbf{x}_i$, and C is a regularization parameter that balances training error and model complexity.

We use $\mathbf{w}$ to identify the most informative features for a partition. Note that a feature with a large positive weight is strongly indicative of the positive class, whereas a feature with a large negative weight is strongly indicative of the negative class. In other words, the most informative features are those with large absolute weight values. We exploit this observation and identify the most informative features for a partition by (1) training an SVM classifier[4] on the partition, where data points in the same cluster belong to the same class; (2) sorting the features according to the SVM-learned feature weights; and (3) generating two ranked lists of informative features using the top and bottom 100 features, respectively.

Given the ranked lists generated for each of the four partitions, the user will select one of the partitions/dimensions as most relevant to sentiment by inspecting as many features in the ranked lists as needed. After picking the most relevant dimension, the user will label one of the two feature lists associated with this dimension as POSITIVE and the other as NEGATIVE. Since each feature list represents one of the clusters, the cluster associated with the positive list is labeled POSITIVE and

[3]The first eigenvector is not used because it is a constant vector, meaning that it cannot be used to partition the data.

[4]All the SVM classifiers in this paper are trained using the SVM^{light} package (Joachims, 1999), with the learning parameters set to their default values.

the cluster associated with the negative list is labeled NEGATIVE.

In comparison to existing user feedback mechanisms for assisting a clustering algorithm, ours requires comparatively little human intervention: we only require that the user select a dimension by examining a small number of features, as opposed to having the user construct the feature space or identify clusters that need to be merged or split as is required with other methods.

Step 3: Identify the unambiguous reviews
There is a caveat, however. As mentioned in the introduction, many reviews contain both positive and negative sentiment-bearing words. These ambiguous reviews are more likely to be clustered incorrectly than their unambiguous counterparts. Now, since the ranked lists of features are derived from the partition, the presence of these ambiguous reviews can adversely affect the identification of informative features using MMFR. As a result, we remove the ambiguous reviews before deriving informative features from a partition.

We employ a simple method for identifying unambiguous reviews. In the computation of eigenvalues, each data point factors out the orthogonal projections of each of the other data points with which they have an affinity. Ambiguous data points receive the orthogonal projections from both the positive and negative data points, and hence they have near zero values in the pivot eigenvectors. We exploit this important information. The basic idea is that the data points with near zero values in the eigenvectors are more ambiguous than those with large absolute values. As a result, we posit 250 reviews from each cluster whose corresponding values in the eigenvector are farthest away from zero as unambiguous, and induce the ranked list of features only from the resulting 500 unambiguous reviews.[5]

Step 4: Cluster along the selected dimension
Finally, we employ the 2-means algorithm to cluster all the reviews along the dimension (i.e., the eigenvector) selected by the user, regardless of whether a review is ambiguous or not.

[5]Note that 500 is a somewhat arbitrary choice. Underlying this choice is our assumption that a fraction of the reviews is unambiguous. As we will see in the evaluation section, these 500 reviews can be classified with a high accuracy; consequently, the features induced from the resulting clusters are also of high quality. Additional experiments reveal that the list of top-ranking features does not change significantly when induced from a smaller number of unambiguous reviews.

4 Evaluation

4.1 Experimental Setup

Datasets. We use five sentiment classification datasets, including the widely-used movie review dataset [MOV] (Pang et al., 2002) as well as four datasets containing reviews of four different types of products from Amazon [books (BOO), DVDs (DVD), electronics (ELE), and kitchen appliances (KIT)] (Blitzer et al., 2007). Each dataset has 2000 labeled reviews (1000 positives and 1000 negatives). To illustrate the difference between topic-based clustering and sentiment-based clustering, we will also show topic-based clustering results on POL, a dataset created by taking all the documents from two sections of 20 Newsgroups, namely, `sci.crypt` and `talks.politics`.

To preprocess a document, we first tokenize and downcase it, and then represent it as a vector of unigrams, using frequency as presence. In addition, we remove from the vector punctuation, numbers, words of length one, and words that occur in only a single review. Following the common practice in the information retrieval community, we also exclude words with high document frequency, many of which are stopwords or domain-specific general-purpose words (e.g., "movies" in the movie domain). A preliminary examination of our evaluation datasets reveals that these words typically comprise 1–2% of a vocabulary. The decision of exactly how many terms to remove from each dataset is subjective: a large corpus typically requires more removals than a small corpus. To be consistent, we simply sort the vocabulary by document frequency and remove the top 1.5%.

Evaluation metrics. We employ two evaluation metrics. First, we report results in terms of the accuracy achieved on the 2000 labeled reviews for each dataset. Second, following Kamvar et al. (2003), we evaluate the clusters produced by our approach against the gold-standard clusters using the Adjusted Rand Index (ARI). ARI ranges from –1 to 1; better clusterings have higher ARI values.

4.2 Baseline Systems

Clustering using the second eigenvector only. As our first baseline, we adopt Shi and Malik's approach and cluster the reviews using only the second eigenvector, $\mathbf{e}_2$, as described in Section 2.2. Results on POL and the five sentiment datasets are

System Variation	Accuracy						Adjusted Rand Index					
	POL	MOV	KIT	BOO	DVD	ELE	POL	MOV	KIT	BOO	DVD	ELE
Baseline: 2nd eigenvector	93.7	70.9	69.7	58.9	55.3	50.8	0.76	0.17	0.15	0.03	0.01	0.01
Baseline: m eigenvectors	95.9	59.3	63.2	60.1	62.5	63.8	0.84	0.03	0.07	0.04	0.06	0.08
Our approach	93.7	70.9	69.7	69.5	70.8	65.8	0.76	0.17	0.15	0.15	0.17	0.10

Table 1: Results in terms of accuracy and Adjusted Rand Index for the six datasets.

shown in row 1 of Table 1.[6] As we can see, this baseline achieves an accuracy of 90% on POL, but a much lower accuracy (of 50–70%) on the sentiment datasets. The same performance trend can be observed with ARI. These results provide support for the claim that sentiment-based clustering is more difficult than topic-based clustering.

In addition, it is worth noting that the baseline achieves much lower accuracies and ARI values on BOO, DVD, and ELE than on the remaining two sentiment datasets. Since $\mathbf{e}_2$ captures the most prominent dimension, these results suggest that sentiment dimension is not the most prominent dimension in these three datasets. In fact, this is intuitively plausible. For instance, in the book domain, positive book reviews typically contain a short description of the content, with the reviewer only briefly expressing her sentiment somewhere in the review. Similarly for the electronics domain: electronic product reviews are typically aspect-oriented, with the reviewer talking about the pros and cons of each aspect of the product (e.g., battery, durability). Since the reviews are likely to contain both positive and negative sentiment-bearing words, the sentiment-based clustering is unlikely to be captured by $\mathbf{e}_2$.

Clustering using top five eigenvectors. As our second baseline, we represent each data point using the top five eigenvectors (i.e., $\mathbf{e}_1$ through $\mathbf{e}_5$), and cluster them using 2-means in this 5-dimensional space, as described in Section 2.2. Hence, this can be thought of as an "ensemble" approach, where the clustering decision is collectively made by the five eigenvectors.

Results are shown in row 2 of Table 1. In comparison to the first baseline, we see improvements in accuracy and ARI for the three datasets on which the first baseline performs poorly (i.e., BOO, DVD, and ELE), with the most drastic improvement observed on ELE. On the other hand, performance on the remaining two sentiment datasets deteriorates. These results can be attributed to the fact that for BOO, DVD, and ELE, $\mathbf{e}_2$ does not capture the sentiment dimension, but since some other eigenvector in the ensemble does, we see improvements. On the other hand, $\mathbf{e}_2$ has already captured the sentiment dimension in MOV and KIT; as a result, employing additional dimensions, which may not be sentiment-related, may only introduce noise into the computation of the similarities between the reviews.

[6] Owing to the randomness in the choice of seeds for 2-means, these and all other experimental results involving 2-means are averaged over ten independent runs.

4.3 Our Approach

Human experiments. Unlike the two baselines, our approach requires users to specify which of the four dimensions (defined by the second through fifth eigenvectors) are most closely related to sentiment by inspecting a set of features derived from the unambiguous reviews for each dimension using MMFR. To better understand how easy it is for a human to select the desired dimension given the features, we performed the experiment independently with five humans (all of whom are computer science graduate students not affiliated with this research) and computed the agreement rate.

More specifically, for each dataset, we showed each human judge the top 100 features for each cluster according to MMFR (see Tables 4–6 for a snippet). In addition, we informed them of the intended dimension: for example, for POL, the judge was told that the intended clustering is Politics vs. Science. Also, if she determined that more than one dimension was relevant to the intended clustering, she was instructed to rank these dimensions in terms of their degree of relevance, where the most relevant one would appear first in the list.

The dimensions (expressed in terms of the IDs of the eigenvectors) selected by each of the five judges for each dataset are shown in Table 2. The agreement rate (shown in the last row of the table) was computed based on only the highest-ranked dimension selected by each judge. As we can see, perfect agreement is achieved for four of the five sentiment datasets, and for the remaining two datasets, near-perfect agreement is achieved.

Judge	POL	MOV	KIT	BOO	DVD	ELE
1	2,3,4	2	2	4	3	3
2	2,4	2	2	4	3	3
3	4	2,4	4	4	3	3
4	2,3	2	2	4	3	3,4
5	2	2	2	4	3	3
Agr	80%	100%	80%	100%	100%	100%

Table 2: Human agreement rate.

	POL	MOV	KIT	BOO	DVD	ELE
Acc	99.8	87.0	87.6	86.2	87.4	77.6

Table 3: Accuracies on unambiguous documents.

These results together with the fact that it took 5–6 minutes to identify the relevant dimension, indicate that asking a human to determine the intended dimension based on solely the "informative" features is a viable task.

Clustering results. Next, we cluster all 2000 documents for each dataset using the dimension selected by the majority of the human judges. The clustering results are shown in row 3 of Table 1. In comparison to the better baseline for each dataset, we see that our approach performs substantially better on BOO, DVD and ELE, at almost the same level on MOV and KIT, but slightly worse on POL. Note that the improvements observed for BOO, DVD and ELE can be attributed to the failure of $\mathbf{e}_2$ to capture the sentiment dimension. Perhaps most importantly, by exploiting human feedback, our approach has achieved more stable performance across the datasets than the baselines, with accuracies ranging from 65.8% to 93.7% and ARI ranging from 0.10 to 0.76.

Role of unambiguous documents. Recall that the features with the largest MMFR were computed from the unambiguous documents only. To get an intuitive understanding of the role of unambiguous documents in our approach, we show in Table 3 the accuracy when the unambiguous documents in each dataset were clustered using the eigenvector selected by the majority of the judges. As we can see, the accuracy of each dataset is higher than the corresponding accuracy shown in row 3 of Table 1. In fact, an accuracy of more than 85% was achieved on all but one dataset. This suggests that our method of identifying unambiguous documents is useful.

Note that it is crucial to be able to achieve a high accuracy on the unambiguous documents: if clustering accuracy is low, the features induced from the clusters may not be an accurate representation of the corresponding dimension, and the human judge may have a difficult time identifying the intended dimension. In fact, some human judges reported difficulty in identifying the correct dimension for the ELE dataset, and this can be attributed in part to the low accuracy achieved on the unambiguous documents.

Features as summary. Recall that the method we proposed represents each dimension with a small number of features and asks a user to select the desired dimension by inspecting the corresponding feature lists. In other words, each feature list serves as a "summary" of its corresponding dimension, and inspecting the features induced for each dimension can give us insights into the different dimensions of a dataset. Hence, if a user is not sure how she wants the data points to be clustered (due to lack of knowledge of the data, for instance), our automatically induced features may serve as an overview of the different dimensions of the data. To better understand whether these features can indeed provide a user with additional useful information about a dataset, we show in Tables 4–6 the top ten features induced for each cluster and each dimension for the six datasets. As an example, consider the MOV dataset. Inspecting the induced features, we can determine that it has a sentiment dimension ($\mathbf{e}_2$), as well as a humor vs. thriller dimension ($\mathbf{e}_4$). In other words, if we cluster along $\mathbf{e}_2$, we get a sentiment-based clustering; and if we cluster along $\mathbf{e}_4$, we obtain a genre-based (humor vs. thriller) clustering.

User feedback vs. labeled data. Recall that our two baselines are unsupervised, whereas our approach can be characterized as semi-supervised, as it relies on user feedback to select the intended dimension. Hence, it should not be surprising to see that the average clustering performance of our approach is better than that of the baselines.

To do a fairer comparison, we conduct another experiment in which we compare our approach against a semi-supervised sentiment classification system, which uses transductive SVM as the underlying semi-supervised learner. More specifically, the goal of this experiment is to determine how many labeled documents are needed in order for the transductive learner to achieve the same level of performance as our approach. To answer this question, we first give the transductive learner access to the 2000 documents for each dataset as

POL				MOV			
$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$	$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$
$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$
serder	beyer	serbs	escrow	relationship	production	jokes	starts
armenian	arabs	palestinians	serial	son	earth	kids	person
turkey	andi	muslims	algorithm	tale	sequences	live	saw
armenians	research	wrong	chips	husband	aliens	animation	feeling
muslims	israelis	department	ensure	perfect	war	disney	lives
sdpa	tim	bosnia	care	drama	crew	animated	told
argic	uci	live	strong	focus	alien	laughs	happen
davidian	ab	matter	police	strong	planet	production	am
dbd@ura	z@virginia	freedom	omissions	beautiful	horror	voice	felt
troops	holocaust	politics	excepted	nature	evil	hilarious	happened
$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$
sternlight	escrow	standard	internet	worst	sex	thriller	comic
wouldn	sternlight	sternlight	uucp	stupid	romantic	killer	sequences
pgp	algorithm	des	uk	waste	school	murder	michael
crypto	access	escrow	net	bunch	relationship	crime	supporting
algorithm	net	employer	quote	wasn	friends	police	career
isn	des	net	ac	video	jokes	car	production
likely	privacy	york	co	worse	laughs	dead	peter
access	uk	jake	didn	boring	sexual	killed	style
idea	systems	code	ai	guess	cute	starts	latest
cryptograph	pgp	algorithm	mit	anyway	mother	violence	entertaining

Table 4: Top ten features induced for each dimension for the POL and MOV domains. The shaded columns correspond to the dimensions selected by the human judges. $\mathbf{e}_2, \ldots, \mathbf{e}_5$ are the top eigenvectors; $\mathbf{C}_1$ and $\mathbf{C}_2$ are the clusters.

BOO				ELE			
$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$	$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$
$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$
history	series	loved	must	mouse	music	easy	amazon
must	man	highly	wonderful	cable	really	used	cable
modern	history	easy	old	cables	ipod	card	card
important	character	enjoyed	feel	case	too	fine	recommend
text	death	children	away	red	little	using	dvd
reference	between	again	children	monster	headphones	problems	camera
excellent	war	although	year	picture	hard	fine	fast
provides	seems	excellent	someone	kit	excellent	drive	far
business	political	understand	man	overall	need	computer	printer
both	american	three	made	paid	fit	install	picture
$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$
plot	buy	money	boring	working	worked	money	phone
didn	bought	bad	series	never	problem	worth	off
thought	information	nothing	history	before	never	amazon	worked
boring	easy	waste	pages	phone	item	over	power
got	money	buy	information	days	amazon	return	battery
character	recipes	anything	between	headset	working	years	unit
couldn	pictures	doesn	highly	money	support	much	set
ll	look	already	page	months	months	headphones	phones
ending	waste	instead	excellent	return	returned	sony	range
fan	copy	seems	couldn	second	another	received	little

Table 5: Top ten features induced for each dimension for the BOO and ELE domains. The shaded columns correspond to the dimensions selected by the human judges. $\mathbf{e}_2, \ldots, \mathbf{e}_5$ are the top eigenvectors; $\mathbf{C}_1$ and $\mathbf{C}_2$ are the clusters.

unlabeled data. Next, we randomly sample 50 unlabeled documents and assign them the true label. We then re-train the classifier and compute its accuracy on the 2000 documents. We keep adding more labeled data (50 in each iteration) until it reaches the accuracy achieved by our system. Results of this experiment are shown in Table 7. Owing in the randomness involved in the selection of unlabeled documents, these results are averaged over ten independent runs. As we can see, our

KIT				DVD			
$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$	$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$
$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$	$\mathbf{C}_1$
love	works	really	pan	worth	music	video	money
clean	water	nice	oven	bought	collection	music	quality
nice	clean	works	cooking	series	excellent	found	video
size	work	too	made	money	wonderful	feel	worth
set	ice	quality	pans	season	must	bought	found
kitchen	makes	small	better	fan	loved	workout	version
easily	thing	sturdy	heat	collection	perfect	daughter	picture
sturdy	need	little	cook	music	highly	recommend	waste
recommend	keep	think	using	tv	makes	our	special
price	best	item	clean	thought	special	disappointed	sound
$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$	$\mathbf{C}_2$
months	price	ve	love	young	worst	series	saw
still	item	years	coffee	between	money	cast	watched
back	set	love	too	actors	thought	fan	loved
never	ordered	never	recommend	men	boring	stars	enjoy
worked	amazon	clean	makes	cast	nothing	original	whole
money	gift	months	over	seems	minutes	comedy	got
did	got	over	size	job	waste	actors	family
amazon	quality	pan	little	beautiful	saw	worth	series
return	received	been	maker	around	pretty	classic	season
machine	knives	pans	cup	director	reviews	action	liked

Table 6: Top ten features induced for each dimension for the KIT and DVD domains. The shaded columns correspond to the dimensions selected by the human judges. $\mathbf{e}_2$, . . ., $\mathbf{e}_5$ are the top eigenvectors; $\mathbf{C}_1$ and $\mathbf{C}_2$ are the clusters.

	POL	MOV	KIT	BOO	DVD	ELE
# labels	400	150	200	350	350	200

Table 7: Transductive SVM results.

user feedback is equivalent to the effort of hand-annotating 275 documents per dataset on average.

Multiple relevant dimensions. As seen from Table 2, some human judges selected more than one dimension for some datasets (e.g., 2,3,4 for POL; 2,4 for MOV; and 3,4 for ELE). However, we never took into account these "extra" dimensions in our previous experiments. To better understand whether these extra dimensions can help improve accuracy and ARI, we conduct another experiment in which we apply 2-means to cluster the documents in a space that is defined by all of the selected dimensions. The final accuracy turns out to be 95.9%, 70.9%, and 67.5% for POL, MOV, and ELE respectively, which is considerably better than using only the optimal dimension and suggests that the extra dimensions contain useful information.

5 Conclusions

Unsupervised clustering algorithms typically group objects along the most prominent dimension, in part owing to their objective of simultaneously maximizing inter-cluster similarity and intra-cluster dissimilarity. Hence, if the user's intended clustering dimension is not the most prominent dimension, these unsupervised clustering algorithms will fail miserably. To address this problem, we proposed to integrate a novel user feedback mechanism into a spectral clustering algorithm, which allows us to mine the intended, possibly hidden, dimension of the data and produce the desired clustering. This mechanism differs from competing methods in that it requires very limited feedback: to select the intended dimension, the user only needs to inspect a small number of features. We demonstrated its viability via a set of human and automatic experiments with unsupervised sentiment classification, obtaining promising results.

In future work, we plan to explore several extensions to our proposed method. First, we plan to use our user-feedback method in combination with existing methods (e.g., Bekkerman et al. (2007)) for improving its performance. For instance, instead of having the user construct a relevant feature space from scratch, she can simply extend the set of informative features identified for the user-selected dimension. Second, since none of the steps in our method is specifically designed for sentiment classification, we plan to apply it to other non-topic-based text classification tasks.

Acknowledgments

We thank the three anonymous reviewers for their invaluable comments on an earlier draft of the paper. This work was supported in part by NSF Grant IIS-0812261.

References

Maria-Florina Balcan and Avrim Blum. 2008. Clustering with interactive feedback. In *Proceedings of ALT*, pages 316–328.

Carmen Banea, Rada Mihalcea, Janyce Wiebe, and Samer Hassan. 2008. Multilingual subjectivity analysis using machine translation. In *Proceedings of EMNLP*, pages 127–135.

Ron Bekkerman, Hema Raghavan, James Allan, and Koji Eguchi. 2007. Interactive clustering of text collections according to a user-specified criterion. In *Proceedings of IJCAI*, pages 684–689.

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

Deng Cai, Xiaofei He, and Jiawei Han. 2005. Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1624–1637.

Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. 2004. Kernel k-means, spectral clustering and normalized cuts. In *Proceedings of KDD*, pages 551–556.

Mark Dredze and Koby Crammer. 2008. Active learning with confidence. In *Proceedings of ACL-08:HLT Short Papers (Companion Volume)*, pages 233–236.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In Bernhard Scholkopf and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 44–56. MIT Press.

Sepandar Kamvar, Dan Klein, and Chris Manning. 2003. Spectral learning. In *Proceedings of IJCAI*, pages 561–566.

Ravi Kannan, Santosh Vempala, and Adrian Vetta. 2004. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497–515.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of COLING*, pages 1367–1373.

Bing Liu, Xiaoli Li, Wee Sun Lee, and Philip S. Yu. 2004. Text classification by labeling words. In *Proceedings of AAAI*, pages 425–430.

Tony Mullen and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of EMNLP*, pages 412–418.

Andrew Ng, Michael Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *Advances in NIPS 14*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86.

Hema Raghavan and James Allan. 2007. An interactive algorithm for asking and incorporating feature feedback into support vector machines. In *Proceedings of SIGIR*, pages 79–86.

Dan Roth and Kevin Small. 2009. Interactive feature space construction using semantic information. In *Proceedings of CoNLL*, pages 66–74.

Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

Peter Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the ACL*, pages 417–424.

Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. 2001. Constrained k-means clustering with background knowledge. In *Proceedings of ICML*, pages 577–584.

Xiaojun Wan. 2008. Using bilingual knowledge and ensemble techniques for unsupervised Chinese sentiment analysis. In *Proceedings of EMNLP*, pages 553–561.

Yair Weiss. 1999. Segmentation using eigenvectors: A unifying view. In *Proceedings of ICCV*, pages 975–982.

Yiming Yang and Jan Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of ICML*, pages 412–420.

Adapting a Polarity Lexicon using Integer Linear Programming for Domain-Specific Sentiment Classification

Yejin Choi and Claire Cardie
Department of Computer Science
Cornell University
Ithaca, NY 14853
{ychoi,cardie}@cs.cornell.edu

Abstract

Polarity lexicons have been a valuable resource for sentiment analysis and opinion mining. There are a number of such lexical resources available, but it is often suboptimal to use them as is, because general purpose lexical resources do not reflect domain-specific lexical usage. In this paper, we propose a novel method based on integer linear programming that can adapt an existing lexicon into a new one to reflect the characteristics of the data more directly. In particular, our method collectively considers the relations among words and opinion expressions to derive the most likely polarity of each lexical item (*positive, neutral, negative*, or *negator*) for the given domain. Experimental results show that our lexicon adaptation technique improves the performance of fine-grained polarity classification.

1 Introduction

Polarity lexicons have been a valuable resource for sentiment analysis and opinion mining. In particular, they have been an essential ingredient for fine-grained sentiment analysis (e.g., Kim and Hovy (2004), Kennedy and Inkpen (2005), Wilson et al. (2005)). Even though the polarity lexicon plays an important role (Section 3.1), it has received relatively less attention in previous research. In most cases, polarity lexicon construction is discussed only briefly as a preprocessing step for a sentiment analysis task (e.g., Hu and Liu (2004), Moilanen and Pulman (2007)), but the effect of different alternative polarity lexicons is not explicitly investigated. Conversely, research efforts that focus on constructing a general purpose polarity lexicon (e.g., Takamura et al. (2005), Andreevskaia and Bergler (2006), Esuli and Sebastiani (2006), Rao and Ravichandran (2009)) generally evaluate the lexicon in isolation from any potentially relevant NLP task, and it is unclear how the new lexicon might affect end-to-end performance of a concrete NLP application.

It might even be unrealistic to expect that there can be a general-purpose lexical resource that can be effective across *all* relevant NLP applications, as general-purpose lexicons will not reflect domain-specific lexical usage. Indeed, Blitzer et al. (2007) note that the polarity of a particular word can carry opposite sentiment depending on the domain (e.g., Andreevskaia and Bergler (2008)).

In this paper, we propose a novel method based on integer linear programming to adapt an existing polarity lexicon into a new one to reflect the characteristics of the data more directly. In particular, our method considers the relations among words and opinion expressions collectively to derive the most likely polarity of each word for the given domain.

Figure 1 depicts the key insight of our approach using a bipartite graph. On the left hand side, each node represents a word, and on the right hand side, each node represents an opinion expression. There is an edge between a word w_i and an opinion expression e_j, if the word w_i appears in the expression e_j. We assume the possible polarity of each expression is one of the following three values: {*positive, neutral, negative*}, while the possible polarity of each word is one of: {*positive*, *neutral*, *negative* or *negator*}. Strictly speaking, *negator* is not a value for polarity, but we include them in our lexicon, because *valence shifters* or *negators* have been shown to play an important role for sentiment analysis (e.g., Polanyi and Zaenen (2004), Moilanen and Pulman (2007), Choi and Cardie (2008)).

Typically, the ultimate goal of the sentiment analysis task is to determine the expression-level (or sentiment/ document-level) polarities, rather

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 590–598,
Singapore, 6-7 August 2009.

than the correct word-level polarities with respect to the domain. Therefore, word-level polarities can be considered as latent information. In this paper, we show how we can improve the word-level polarities of a general-purpose polarity lexicon by utilizing the expression-level polarities, and in return, how the adapted word-level polarities can improve the expression-level polarities.

In Figure 1, there are two types of relations we could exploit when adapting a general-purpose polarity lexicon into a domain-specific one. The first are word-to-word relations within each expression. That is, if we are not sure about the polarity of a certain word, we can still make a guess based on the polarities of other words within the same expression and knowledge of the polarity of the expression. The second type of relations are word-to-expression relations: e.g., some words appear in expressions that take on a variety of polarities, while other words are associated with expressions of one polarity class or another.

In relation to previous research, analyzing word-to-word (intra-expression) relations is most related to techniques that determine expression-level polarity *in context* (e.g., Wilson et al. (2005)), while exploring word-to-expression (inter-expression) relations has connections to techniques that employ more of a global-view of corpus statistics (e.g., Kanayama and Nasukawa (2006)).[1]

While most previous research exploits only one or the other type of relation, we propose a unified method that can exploit both types of semantic relation, while adapting a general purpose polarity lexicon into a domain specific one. We formulate our lexicon adaptation task using integer linear programming (ILP), which has been shown to be very effective when solving problems with complex constraints (e.g., Roth and Yih (2004), Denis and Baldridge (2007)). And the word-to-word and word-to-expression relations discussed above can be encoded as soft and hard constraints in ILP. Unfortunately, one class of constraint that we would like to encode (see Section 2) will require an exponentially many number of constraints when grounded into an actual ILP problem. We therefore propose an approximation scheme to make the problem more practically solvable.

We evaluate the effect of the adapted lexicon in the context of a concrete NLP task: expression-level polarity classification. Experimental results show that our lexicon adaptation technique improves the accuracy of two competitive expression-level polarity classifiers from 64.2% - 70.4% to 67.0% - 71.2%..

[1] In case of document-level polarity classification, word-to-expression relations correspond to word-to-document relations.

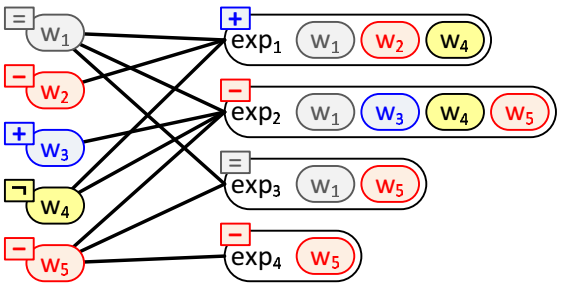


Figure 1: The relations among words and expressions. + indicates positive, - indicates negative, = indicates neutral, and ¬ indicates a negator.

2 An Integer Linear Programming Approach

In this section, we describe how we formulate the lexicon adaptation task using integer linear programming. Before we begin, we assume that we have a general-purpose polarity lexicon $\mathcal{L}$, and a polarity classification algorithm $f(e_l, \mathcal{L})$, that can determine the polarity of the opinion expression e_l based on the words in e_l and the initial lexicon $\mathcal{L}$. The polarity classification algorithm $f(\cdot)$ can be either a heuristic-based one, or a machine-learning based one – we consider it as a black box for now.

Constraints for word-level polarities: For each word x_i, we define four binary variables: $x_i^+, x_i^=, x_i^-, x_i^\neg$ to represent positive, neutral, negative polarity, and negators respectively. If $x_i^\delta = 1$ for some $\delta \in \{+, =, -, \neg\}$, then the word x_i has the polarity δ. The following inequality constraint states that at least one polarity value must be chosen for each word.

$$x_i^+ + x_i^= + x_i^- + x_i^\neg >= 1 \qquad (1)$$

If we allow only one polarity per word, then the above inequality constraint should be modified as an equality constraint. Although most words tend to associate with a single polarity, some can take on more than one polarity. In order to capture this observation, we introduce an auxiliary binary variable α_i for each word x_i. Then the next inequality

constraint states that at most two polarities can be chosen for each word.

$$x_i^+ + x_i^= + x_i^- + x_i^\neg <= 1 + \alpha_i \quad (2)$$

Next we introduce the initial part of our objective function.

$$\text{maximize} \sum_i \Big(w_i^+ x_i^+ + w_i^= x_i^= + w_i^- x_i^- + w_i^\neg x_i^\neg - w_\alpha \alpha_i \Big) + \cdots \quad (3)$$

For the auxiliary variable α_i, we apply a constant weight w_α to discourage ILP from choosing more than one polarity for each word. We can allow more than two polarities for each word, by adding extra auxiliary variables and weights. For each variable x_i^δ, we define its weight w_i^δ, which indicates how likely it is that word x_i carries the polarity δ. We define the value of w_i^δ using two different types of information as follows:

$$w_i^\delta := {}^{\mathcal{L}}w_i^\delta + {}^{\mathcal{C}}w_i^\delta$$

where ${}^{\mathcal{L}}w_i^\delta$ is the degree of polarity δ for word x_i determined by the general-purpose polarity lexicon $\mathcal{L}$, and ${}^{\mathcal{C}}w_i^\delta$ is the degree of polarity δ determined by the corpus statistics as follows:[2]

$${}^{\mathcal{C}}w_i^\delta := \frac{\#\text{ of } x_i \text{ in expressions with polarity } \delta}{\#\text{ of } x_i \text{ in the corpus } \mathcal{C}}$$

Note that the occurrence of word x_i in an expression e_j with a polarity δ does not necessarily mean that the polarity of x_i should also be δ, as the interpretation of the polarity of an expression is more than just a linear sum of the word-level polarities (e.g., Moilanen and Pulman (2007)). Nonetheless, not all expressions require a complicated inference procedure to determine their polarity. Therefore, ${}^{\mathcal{C}}w_i^\delta$ still provides useful information about the likely polarity of each word based on the corpus statistics.

From the perspective of Chomskyan linguistics, the weights ${}^{\mathcal{L}}w_i^\delta$ based on the prior polarity from the lexicon can be considered as having a "competence" component , while ${}^{\mathcal{C}}w_i^\delta$ derived from the corpus counts can be considered as a "performance" component (Noam Chomsky (1965)).

[2]If a word x_i is in an expression that is not an opinion, then we count it as an occurrence with neutral polarity.

Constraints for content-word negators: Next we describe a constraint that exploits knowledge of the typical distribution of *content-word negators* in natural language. Content-word negators are words that are not function words, but act semantically as negators (Choi and Cardie, 2008).[3] Although it is possible to artificially construct a very convoluted sentence with lots of negations, it is unlikely for multiple layers of negations to appear very often in natural language (Pickett et al. (1996)). Therefore, we allow at most one content-word negator for each expression e_l. Because we do not restrict the number of function-word negators, our constraint still gives room for multiple layers of negations.

$$\sum_{i \in \mu(e_l)} x_i^\neg <= 1 \quad (4)$$

In the above constraint, $\mu(e_l)$ indicates the set of indices of content words appearing in e_l. For instance, if $i \in \mu(e_l)$, then x_i appears in e_l. This constraint can be polished further to accommodate longer expressions where multiple content-word negators are more likely to appear, by adding a separate constraint with a sliding window.

Constraints for expression-level polarities: Before we begin, we introduce $\pi(e_l)$ that will be used often in the remaining section. For each expression e_l, we define $\pi(e_l)$ to be the set of content words appearing in e_l, together with the most likely polarity proposed by a general-purpose polarity lexicon $\mathcal{L}$. For instance, if $x_i^+ \in \pi(e_l)$, then the polarity of word x_i is $+$ according to $\mathcal{L}$.

Next we encode constraints that consider expression-level polarities. If the polarity classification algorithm $f(e_l, \mathcal{L})$ makes an incorrect prediction for e_l using the original lexicon $\mathcal{L}$, then we need to encourage ILP to fix the error by suggesting different word-level polarities. We capture this idea by the following constraint:

$$\sum_{x_i^\delta \in \pi(e_l)} x_i^\delta <= |\pi(e_l)| - 1 + \beta_l \quad (5)$$

The auxiliary binary variable β_l is introduced for each e_l so that the assignment $\pi(e_l)$ does not have to be changed if paying for the cost w_β in the objective function. (See equation (10).) That is, suppose the ILP solver assigns '1' to all variables

[3]Examples of content-word negators are *destroy*, *eliminate*, *prevent* etc.

in $\phi(e_l)$, (which corresponds to keeping the original lexicon as it is for all words in the given expression e_l), then the auxiliary variable β_l must be also set as '1' in order to satisfy the constraint (5). Because β_l is associated with a negative weight in the objective function, doing so will act against maximizing the objective function. This way, we discourage the ILP solver to preserve the original lexicon as it is.

To verify the constraint (5) further, suppose that the ILP solver assigns '1' for all variables in $\phi(e_l)$ except for one variable. (Notice that doing so corresponds to proposing a new polarity for one of the words in the given expression e_l.) Then the constraint (5) will hold regardless of whether the ILP solver assigns '0' or '1' to β_l. Because β_l is associated with a negative weight in the objective function, the ILP solver will then assign '0' to β_l to maximize the objective function. In other words, we encourage the ILP solver to modify the original lexicon for the given expression e_l.

We use this type of *soft* constraint in order to cope with the following two noise factors: first, it is possible that some annotations are noisy. Second, $f(e_l, \mathcal{L})$ is not perfect, and might not be able to make a correct prediction even with the correct word-level polarities.

Next we encode a constraint that is the opposite of the previous one. That is, if the polarity classification algorithm $f(e_l, \mathcal{L})$ makes a correct prediction on e_l using the original lexicon $\mathcal{L}$, then we encourage ILP to keep the original word-level polarities for words in e_l.

$$\sum_{x_i^\delta \in \pi(e_l)} x_i^\delta \ >= \ |\pi(e_l)| - |\pi(e_l)|\beta_l \qquad (6)$$

Interpretation of constraint (6) with the auxiliary binary variable β_l is similar to that of constraint (5) elaborated above.

Notice that in equation (5), we encouraged ILP to fix the current lexicon $\mathcal{L}$ for words in e_l, but we have not specified the consequence of a modified lexicon ($\mathcal{L}'$) in terms of expression-level polarity classification $f(e_l, \mathcal{L}')$. Certain changes to $\mathcal{L}$ might not fix the prediction error for e_l, and those might even cause extra incorrect predictions for other expressions. Then it would seem that we need to replicate constraints (5) & (6) for all permutations of word-level polarities. However, doing so would incur exponentially many number of constraints ($4^{|e_l|}$) for each expression.[4]

To make the problem more practically solvable, we only consider changes to the lexicon that are within edit-one distance with respect to $\pi(e_l)$. More formally, let us define $\pi'(e_l)$ to be the set of content words appearing in e_l, together with the most likely polarity proposed by a modified polarity lexicon $\mathcal{L}'$. Then we need to consider all $\pi'(e_l)$ such that $|\pi'(e_l) \cap \pi(e_l)| = |\pi(e_l)| - 1$. There are $(4-1)|e_l|$ number of different $\pi'(e_l)$, and we index them as $\pi'_k(e_l)$. We then add following constraints similarly as equation (5) & (6):

$$\sum_{x_i^\delta \in \pi'_k(e_l)} x_i^\delta \ <= \ |\pi'_k(e_l)| - 1 + \beta_{(l,k)} \qquad (7)$$

if the polarity classification algorithm $f(\cdot)$ makes an incorrect prediction based on $\pi'_k(e_l)$. And,

$$\sum_{x_i^\delta \in \pi'_k(e_l)} x_i^\delta \ >= \ |\pi'_k(e_l)| - |\pi'_k(e_l)|\beta_{(l,k)} \qquad (8)$$

if the polarity classification algorithm $f(\cdot)$ makes a correct prediction based on $\pi'_k(e_l)$. Remember that none of the constraints (5) - (8) enforces assignment $\pi(e_l)$ or $\pi'_k(e_l)$ as a hard constraint. In order to enforce at least one of them to be chosen, we add the following constraint:

$$\sum_{x_i^\delta \in \pi(e_l)} x_i^\delta \ >= \ |\pi(e_l)| - 1 \qquad (9)$$

This constraint ensures that the modified lexicon $\mathcal{L}'$ is not drastically different from $\mathcal{L}$. Assuming that the initial lexicon $\mathcal{L}$ is a reasonably good one, constraining the search space for $\mathcal{L}'$ will regulate that $\mathcal{L}'$ does not turn into a degenerative one that overfits to the current corpus $\mathcal{C}$.

Objective function: Finally, we introduce our full objective function.

[4]For certain simple polarity classification algorithm $f(e_l, \mathcal{L})$, it is possible to write polynomially many number of constraints. However our approach intends to be more general by treating $f(e_l, \mathcal{L})$ as a black box, so that algorithms that do not factor nicely can also be considered as an option.

$$\begin{aligned}\text{maximize} \sum_i \Big(& w_i^+ x_i^+ + w_i^= x_i^= \\ + & w_i^- x_i^- + w_i^\neg x_i^\neg \\ - & w_\alpha \alpha_i \Big) \\ -\sum_l & w_\beta \rho_l \beta_l \\ -\sum_{l,k} & w_\beta \rho_{(l,k)} \beta_{(l,k)} \end{aligned} \quad (10)$$

We have already described the first part of the objective function (equation (3)), thus we only describe the last two terms here. w_β is defined similarly as w_α; it is a constant weight that applies for any auxiliary binary variable β_l and $\beta_{(l,k)}$.

We further define ρ_l and $\rho_{(l,k)}$ as secondary weights, or *amplifiers* to adjust the constant weight w_β. To enlighten the motivation behind the amplifiers ρ_l and $\rho_{(l,k)}$, we bring out the following observations:

1. Among the incorrect predictions for expression-level polarity classification, some are more incorrect than the other. For instance, classifying positive class to negative class is more wrong than classifying positive class to neutral class. Therefore, the cost of not fixing very incorrect predictions should be higher than the cost of not fixing less incorrect predictions. (See [R2] and [R3] in Table 1.)

2. If the current assignment $\pi(e_l)$ for expression e_l yields a correct prediction using the classifier $y(e_l, \mathcal{L})$, then there is not much point in changing $\mathcal{L}$ to $\mathcal{L}'$, even if $y(e_l, \mathcal{L}')$ also yields a correct prediction. In this case, we would like to assign slightly higher confidence in the original lexicon $\mathcal{L}$ then the new one $\mathcal{L}'$. (See [R1] in Table 1.)

3. Likewise, if the current assignment $\pi(e_l)$ for expression e_l yields an incorrect prediction using the classifier $y(e_l, \mathcal{L})$, then there is not much point in changing $\mathcal{L}$ to $\mathcal{L}'$, if $y(e_l, \mathcal{L}')$ also yields an equally incorrect prediction. Again we assign slightly higher confidence in the original lexicon $\mathcal{L}$ than the new one $\mathcal{L}'$ in such cases. (Compare each row in [R2] with a corresponding row in [R3] in Table 1.)

[R1]	If $\pi(e_l)$ correct	ρ_l	$\leftarrow 1.5$
	If $\pi'_k(e_l)$ correct	$\rho_{(l,k)}$	$\leftarrow 1.0$
[R2]	If $\pi(e_l)$ very incorrect	ρ_l	$\leftarrow 1.0$
	If $\pi(e_l)$ less incorrect	ρ_l	$\leftarrow 0.5$
[R3]	If $\pi'_k(e_l)$ very incorrect	$\rho_{(l,k)}$	$\leftarrow 1.5$
	If $\pi'_k(e_l)$ less incorrect	$\rho_{(l,k)}$	$\leftarrow 1.0$

Table 1: The value of amplifiers ρ_l and $\rho_{(l,k)}$.

To summarize, for correct predictions, the degree of ρ determines the degree of cost of (undesirably) altering the current lexicon for e_l. For incorrect predictions, the degree of ρ determines the degree of cost of not fixing the current lexicon for e_l.

3 Experiments

In the experiment section, we seek for answers for the following questions:

Q1 What is the effect of a polarity lexicon on the expression-level polarity classification task? In particular, is it useful when using a machine learning technique that might be able to learn the necessary polarity information just based on the words in the training data, without consulting a dictionary? (Section 3.1)

Q2 What is the effect of an adapted polarity lexicon on the expression-level polarity classification task? (Section 3.2)

Notice that we include the *neutral* polarity in the polarity classification. It makes our task much harder (e.g., Wilson et al. (2009)) than those that assume inputs are guaranteed to be either strongly positive or negative (e.g., Pang et al. (2002), Choi and Cardie (2008)). But in practice, one cannot expect that a given input is strongly polar, as automatically extracted opinions are bound to be noisy. Furthermore, Wiebe et al. (2005) discuss that some opinion expressions do carry a neutral polarity.

We experiment with the Multi-Perspective Question Answering (MPQA) corpus (Wiebe et al., 2005) for evaluation. It contains 535 newswire documents annotated with phrase-level subjectivity information. We evaluate on all opinion expressions that are known to have high level of inter-annotator agreement. That is, we include opinions with intensity marked as 'medium' or

higher, and exclude those with annotation confidence marked as 'uncertain'. To focus our study on the direct influence of the polarity lexicon upon the sentiment classification task, we assume the boundaries of the expressions are given. However, our approach can be readily used in tandem with a system that extracts opinion expressions (e.g., Kim and Hovy (2005), Breck et al. (2007)). Performance is reported using 10-fold cross-validation on 400 documents, and a separate 135 documents were used as a development set. For the general-purpose polarity lexicon, we expand the polarity lexicon of Wilson et al. (2005) with General Inquirer dictionary as suggested by Choi and Cardie (2008).

We report the performance in two measures: *accuracy* for 3-way classification, and *average error distance*. The reason why we consider *average error distance* is because classifying a positive class into a negative class is worse than classifying a positive class into a neutral one. We define the error distance between 'neutral' class and any other class as 1, while the error distance between 'positive' class and 'negative' class as 2. If a predicted polarity is correct, then the error distance is 0. We compute the error distance of each prediction and take the average over all predictions in the test data.

3.1 Experiment-I: Effect of a Polarity Lexicon

To verify the effect of a polarity lexicon on the expression-level polarity classification task, we experiment with simple classification-based machine learning technique. We use the Mallet (McCallum, 2002) implementation of Conditional Random Fields (CRFs) (Lafferty et al., 2001).[5] To highlight the influence of a polarity lexicon, we compare the performance of CRFs with and without features derived from polarity lexicons.

Features: We encode basic features as words and lemmas for all content words in the given expression. The performance of CRFs using only the basic features are given in the first row of the Table 2. Next we encode features derived from polarity lexicons as follows.

- The output of Vote & Flip algorithm. (Section 3.2 & Figure 2.)

[5]We use the CRF implementation of Mallet (McCallum, 2002) with Markov-order 0, which is equivalent to Maximum Entropy models (Berger et al. (1996)).

	Accuracy	Avg. Error Distance
Without Lexicon	63.9	0.440
With Lexicon	70.4	0.334

Table 2: Effect of a polarity lexicon on expression-level classification using CRFs

- Number of positive, neutral, negative, and negators in the given expression.
- Number of positive (or negative) words in conjunction with number of negators.
- (boolean) Whether the number of positive words dominates negative ones.
- (boolean) Whether the number of negative words dominates positive ones.
- (boolean) None of the above two cases
- Each of the above three boolean values in conjunction with the number of negators.

Results: Table 2 shows the performance of CRFs with and without features that consult the general-purpose lexicon. As expected, CRFs can perform reasonably well (accuracy = 63.9%) even without consulting the dictionary, by learning directly from the data. However, having the polarity lexicon boosts the performance significantly (accuracy = 70.4%), demonstrating that lexical resources are very helpful for fine-grained sentiment analysis. The difference in performance is statistically significant by paired t-test for both accuracy ($p < 0.01$) and average error distance ($p < 0.01$).

3.2 Experiment-II: Adapting a Polarity Lexicon

In this section, we assess the quality of the adapted lexicon in the context of an expression-level polarity classification task. In order to perform the lexicon adaptation via ILP, we need an expression-level polarity classification algorithm $f(e_l, \mathcal{L})$ as described in Section 2. According to Choi and Cardie (2008), voting algorithms that recognize content-word negators achieve a competitive performance, so we will use a variant of it for simplicity. Because none of the algorithms proposed by Choi and Cardie (2008) is designed to handle the neutral polarity, we invent our own version as shown in Figure 2.

For each expression e_i,
  $nPositive \leftarrow$ # of positive words in e_i
  $nNeutral \leftarrow$ # of neutral words in e_i
  $nNegative \leftarrow$ # of negative words in e_i
  $nNegator \leftarrow$ # of negating words in e_i

  if ($nNegator$ % 2 = 0)
    then $fFlipPolarity \leftarrow false$
  else
    then $fFlipPolarity \leftarrow true$

  if ($nPositive > nNegative$) & $\neg$ $fFlipPolarity$
    then Polarity(e_i) $\leftarrow$ $positive$
else if ($nPositive > nNegative$) & $fFlipPolarity$
    then Polarity(e_i) $\leftarrow$ $negative$
else if ($nPositive < nNegative$) & $\neg$ $fFlipPolarity$
    then Polarity(e_i) $\leftarrow$ $negative$
else if ($nPositive < nNegative$) & $fFlipPolarity$
    then Polarity(e_i) $\leftarrow$ $neutral$
else if $nNeutral > 0$
    then Polarity(e_i) $\leftarrow$ $neutral$
else
    then Polarity(e_i) $\leftarrow$ $default\_polarity$ (the most prominent polarity in the corpus)

Figure 2: Vote & Flip Algorithm

It might look a bit complex at first glance, but the intuition is simple. The variable $fFlipPolarity$ determines whether we need to flip the overall majority polarity based on the number of negators in the given expression. If the positive (or negative) polarity words dominate the given expression, and if there is no need to flip the majority polarity, then we take the positive (or negative) polarity as the overall polarity. If the positive (or negative) polarity words dominate the given expression, and if we need to flip the majority polarity, then we take the negative (or neutral) polarity as the overall polarity.

Notice that the result of flipping the negative polarity is *neutral*, not *positive*. In our pilot study, we found that this strategy works better than flipping the negative polarity to positive.[6] Finally, if the number of positive words and the negative words tie, and there is any neutral word, then we assign the neutral polarity. In this case, we don't worry if there is a negator, because flipping a neutral polarity would still result in a neutral polarity. If none of above condition is met, than we default to the most prominent polarity of the data, which is the negative polarity in the MPQA corpus. We name this simple algorithm as Vote & Flip algorithm. The performance is shown in the first row in Table 2.

Next we describe the implementation part of the ILP. For 10 fold-cross validation, we formulate the ILP problem using the training data (360 documents), and then test the effect of the adapted lexicon on the remaining 40 documents. We include only those content words that appeared more than 3 times in the training data. From the pilot test using the development set, we picked the value of w_β as 0.1. We found that having the auxiliary variables α_l which allow more than one polarity per word does not necessarily help with the performance, so we omitted them. We suspect it is because the polarity classifiers we experimented with is not highly capable of disambiguating different lexical usages and select the right polarity for a given context. We use CPLEX integer programming solver to solve our ILP problems. On a machine with 4GHz CPU, it took several minutes to solve each ILP problem.

In order to assess the effect of the adapted lexicon using CRFs, we need to first train the CRFs model. Using the same training set used for the lexicon adaptation would be suboptimal, because the features generated from the adapted lexicon will be unrealistically good in that particular data. Therefore, we prepared a separate training data for CRFs using 135 documents from the development set.

Results: Table 3 shows the comparison of the original lexicon and the adapted lexicon in terms of polarity classification performance using the Vote & Flip algorithm. The adapted lexicon improves the accuracy as well as reducing the average error distance. The difference in performance is statistically significant by paired t-test for both accuracy ($p < 0.01$) and average error distance ($p < 0.01$).

Table 4 shows the comparison of the original lexicon and the adapted lexicon using CRFs. The improvement is not as substantial as that of Vote & Flip algorithm but the difference in performance is also statistically significant for both accuracy ($p = 0.03$) and average error distance ($p = 0.04$).

[6]This finding is not surprising. For instance, if we consider the polarity of *"She did not get hurt much from the accident."*, it can be viewed as neutral; although it is good that one did not hurt much, it is still bad that there was an accident. Hence it gives a mixed feeling, which corresponds to the neutral polarity.

	Accuracy	Avg. Error Distance
Original Lexicon	64.2	0.395
Adapted Lexicon	67.0	0.365

Table 3: Effect of an adapted polarity lexicon on expression-level classification using the Vote & Flip Algorithm

	Accuracy	Avg. Error Distance
Original Lexicon	70.4	0.334
Adapted Lexicon	71.2	0.327

Table 4: Effect of an adapted polarity lexicon on expression-level classification using CRFs

4 Related Work

There are a number of previous work that focus on building polarity lexicons (e.g., Takamura et al. (2005), Kaji and Kitsuregawa (2007), Rao and Ravichandran (2009)). But most of them evaluated their lexicon in isolation from any potentially relevant NLP task, and it is unclear how the new lexicon might affect end-to-end performance of a concrete NLP application. Our work differs in that we try to draw a bridge between general purpose lexical resources and a domain-specific NLP application.

Kim and Hovy (2005) and Banea et al. (2008) present bootstrapping methods to construct a subjectivity lexicon and measure the effect of the new lexicon for sentence-level subjectivity classification. However, their lexicons only tell whether a word is a subjective one, but not the polarity of the sentiment. Furthermore, the construction of lexicon is still an isolated step from the classification task. Our work on the other hand allows the classification task to directly influence the construction of lexicon, enabling the lexicon to be adapted for a concrete NLP application and for a specific domain.

Wilson et al. (2005) pioneered the expression-level polarity classification task using the MPQA corpus. The experimental results are not directly comparable to ours, because Wilson et al. (2005) limit the evaluation only for the words that appeared in their polarity lexicon. Choi and Cardie (2008) also focus on the expression-level polarity classification, but their evaluation setting is not as practical as ours in that they assume the inputs are guaranteed to be either strongly positive or negative.

5 Conclusion

In this paper, we present a novel lexicon adaptation technique based on integer linear programming to reflect the characteristics of the domain more directly. In particular, our method collectively considers the relations among words and opinion expressions to derive the most likely polarity of each lexical item for the given domain. We evaluate the effect of our lexicon adaptation technique in the context of a concrete NLP application: expression-level polarity classification. The positive results from our experiments encourage further research for lexical resource adaptation techniques.

Acknowledgments

This work was supported in part by National Science Foundation Grant BCS-0624277 and by the Department of Homeland Security under ONR Grant N0014-07-1-0152. We also thank the EMNLP reviewers for insightful comments.

References

Alina Andreevskaia and Sabine Bergler. 2008. When Specialists and Generalists Work Together: Overcoming Domain Dependence in Sentiment Tagging. *ACL*

Alina Andreevskaia and Sabine Bergler. 2006. Mining WordNet For a Fuzzy Sentiment: Sentiment Tag Extraction From WordNet Glosses. *EACL*

Carmen Banea, Rada Mihalcea, and JanyceWiebe. 2008. A Bootstrapping Method for Building Subjectivity Lexicons for Languages with Scarce Resources. *LREC*

Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. 1996. A maximum entropy approach to natural language processing. In *Computational Linguistics, 22(1)*

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes, and Blenders: Domain Adaptation for Sentiment Classification. *Association for Computational Linguistics - ACL 2007*

Eric Breck, Yejin Choi and Claire Cardie. 2007. Identifying Expressions of Opinion in Context. In *IJCAI*.

Yejin Choi and Claire Cardie. 2008. Learning with Compositional Semantics as Structural Inference for Subsentential Sentiment Analysis. *EMNLP*

Noam Chomsky. 1965. Aspects of the theory of syntax. Cambridge, MA: MIT Press.

Pascal Denis and Jason Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. *NAACL*

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. In *Proceedings of 5th Conference on Language Resources and Evaluation (LREC),*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2004)*.

Nobuhiro Kaji and Masaru Kitsuregawa. 2007. Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents. In EMNLP-CoNLL.

Hiroshi Kanayama Tetsuya Nasukawa. 2006. Fully Automatic Lexicon Expansion for Domain-oriented Sentiment Analysis. In ACL.

Alistair Kennedy and Diana Inkpen. 2005. Sentiment Classification of Movie and Product Reviews Using Contextual Valence Shifters. In *Proceedings of FINEXIN 2005, Workshop on the Analysis of Informal and Formal Information Exchange during Negotiations.*

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of COLING*.

Soo-Min Kim and Eduard Hovy. 2005. Automatic Detection of Opinion Bearing Words and Sentences. In Companion Volume to the *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP-05)*

John Lafferty, Andrew Kachites McCallum and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*.

Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. http://mallet.cs.umass.edu.

Karo Moilanen and Stephen Pulman. 2007. Sentiment Composition. In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2007).*

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

Joseph Pickett et al. 1996. The American heritage book of English usage: A practical and authoritative guide to contemporary English. Houghton Mifflin Company.

Livia Polanyi and Annie Zaenen. 2004. Contextual lexical valence shifters. In *Exploring Attitude and Affect in Text: Theories and Applications: Papers from the 2004 Spring Symposium, AAAI*.

Delip Rao and Deepak Ravichandran. 2009. Semi-Supervised Polarity Lexicon Induction. In *EACL*.

Dan Roth and Wen-tau Yih. 2004. A Linear Programming Formulation for Global Inference in Natural Language Tasks. In *CoNLL*.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *ACL*.

Janyce Wiebe, Theresa Wilson and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. In *Language Resources and Evaluation (formerly Computers and the Humanities), 39(2-3):165210*.

Theresa Wilson, Janyce Wiebe and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP.*

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2009. Recognizing Contextual Polarity: an exploration of features for phrase-level sentiment analysis. In *Computational Linguistics 35(3).*

Generating High-Coverage Semantic Orientation Lexicons From Overtly Marked Words and a Thesaurus

Saif Mohammad[φ†*], **Cody Dunne**[◇‡], and **Bonnie Dorr**[φ†‡*]
Laboratory for Computational Linguistics and Information Processing[φ]
Human-Computer Interaction Lab[◇]
Institute for Advanced Computer Studies[†]
Department of Computer Science[‡], University of Maryland.
Human Language Technology Center of Excellence[*]
{saif,bonnie}@umiacs.umd.edu and {cdunne}@cs.umd.edu

Abstract

Sentiment analysis often relies on a semantic orientation lexicon of positive and negative words. A number of approaches have been proposed for creating such lexicons, but they tend to be computationally expensive, and usually rely on significant manual annotation and large corpora. Most of these methods use WordNet. In contrast, we propose a simple approach to generate a high-coverage semantic orientation lexicon, which includes both individual words and multi-word expressions, using only a Roget-like thesaurus and a handful of affixes. Further, the lexicon has properties that support the Polyanna Hypothesis. Using the General Inquirer as gold standard, we show that our lexicon has 14 percentage points more correct entries than the leading WordNet-based high-coverage lexicon (SentiWordNet). In an extrinsic evaluation, we obtain significantly higher performance in determining phrase polarity using our thesaurus-based lexicon than with any other. Additionally, we explore the use of visualization techniques to gain insight into the our algorithm beyond the evaluations mentioned above.

1 Introduction

Sentiment analysis involves determining the opinions and private states (beliefs, emotions, speculations, and so on) of the speaker (Wiebe, 1994). It has received significant attention in recent years due to increasing online opinion content and applications in tasks such as automatic product recommendation systems (Tatemura, 2000; Terveen et al., 1997), question answering (Somasundaran et al., 2007; Lita et al., 2005), and summarizing multiple view points (Seki et al., 2004) and opinions (Mohammad et al., 2008a).

A crucial sub-problem is to determine whether positive or negative sentiment is expressed. Automatic methods for this often make use of lexicons of words tagged with positive and negative semantic orientation (Turney, 2002; Wilson et al., 2005; Pang and Lee, 2008). A word is said to have a positive **semantic orientation (SO)** (or **polarity**) if it is often used to convey favorable sentiment or evaluation of the topic under discussion. Some example words that have positive semantic orientation are *excellent, happy, honest*, and so on. Similarly, a word is said to have negative semantic orientation if it is often used to convey unfavorable sentiment or evaluation of the target. Examples include *poor, sad,* and *dishonest*.

Certain semantic orientation lexicons have been manually compiled for English—the most notable being the **General Inquirer (GI)** (Stone et al., 1966).[1] However, the GI lexicon has orientation labels for only about 3,600 entries. The **Pittsburgh subjectivity lexicon (PSL)** (Wilson et al., 2005), which draws from the General Inquirer and other sources, also has semantic orientation labels, but only for about 8,000 words.

Automatic approaches to creating a semantic orientation lexicon and, more generally, approaches for word-level sentiment annotation can be grouped into two kinds: (1) those that rely on manually created lexical resources—most of which use WordNet (Strapparava and Valitutti, 2004; Hu and Liu, 2004; Kamps et al., 2004; Takamura et al., 2005; Esuli and Sebastiani, 2006; An-

[1] http://www.wjh.harvard.edu/ inquirer

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 599–608,
Singapore, 6-7 August 2009.

dreevskaia and Bergler, 2006; Kanayama and Nasukawa, 2006); and (2) those that rely on text corpora (Hatzivassiloglou and McKeown, 1997; Turney and Littman, 2003; Yu and Hatzivassiloglou, 2003; Grefenstette et al., 2004). Many of these lexicons, such as **SentiWordNet (SWN)** (Esuli and Sebastiani, 2006) were created using supervised classifiers and significant manual annotation, whereas others such as the **Turney and Littman lexicon (TLL)** (2003) were created from very large corpora (more than 100 billion words).

In contrast, we propose a computationally inexpensive method to compile a high-coverage semantic orientation lexicon without the use of any text corpora or manually annotated semantic orientation labels. Both of these resources may be used, if available, to further improve results. The lexicon has about twenty times the number of entries in the GI lexicon, and it includes entries for both individual words and common multi-word expressions. The method makes use of a Roget-like thesaurus and a handful of antonym-generating affix patterns. Whereas thesauri have long been used to estimate semantic distance (Jarmasz and Szpakowicz, 2003; Mohammad and Hirst, 2006), the closest thesaurus-based work on sentiment analysis is by Aman and Szpakowicz (2007) on detecting emotions such as happiness, sadness, and anger. We evaluated our thesaurus-based algorithm both intrinsically and extrinsically and show that the semantic orientation lexicon it generates has significantly more correct entries than the state-of-the-art high-coverage lexicon SentiWordNet, and that it has a significantly higher coverage than the General Inquirer and Turney–Littman lexicons.

In Section 2 we examine related work. Section 3 presents our algorithm for creating semantic orientation lexicons. We describe intrinsic and extrinsic evaluation experiments in Section 4, followed by a discussion of the results in Section 5. Additionally, in Section 6 we show preliminary visualizations of how our algorithm forms chains of positive and negative thesaurus categories. Good visualizations are not only effective in presenting information to the user, but also help us better understand our algorithm. Section 7 has our conclusions.

2 Related Work

Pang and Lee (2008) provide an excellent survey of the literature on sentiment analysis. Here we briefly describe the work closest to ours.

Hatzivassiloglou and McKeown (1997) proposed a supervised algorithm to determine the semantic orientation of adjectives. They first generate a graph that has adjectives as nodes. An edge between two nodes indicates either that the two adjectives have the same or opposite semantic orientation. A clustering algorithm partitions the graph into two subgraphs such that the nodes in a subgraph have the same semantic orientation. The subgraph with adjectives that occur more often in text is marked positive and the other negative. They used a 21 million word corpus and evaluated their algorithm on a labeled set of 1336 adjectives (657 positive and 679 negative). Our approach does not require manually annotated semantic orientation entries to train on and is much simpler.

Esuli and Sebastiani (2006) used a supervised algorithm to attach semantic orientation scores to WordNet glosses. They train a set of ternary classifiers using different training data and learning methods. The set of semantic orientation scores of all WordNet synsets is released by the name SentiWordNet.[2] An evaluation of SentiWordNet by comparing orientation scores of about 1,000 WordNet glosses to scores assigned by human annotators is presented in Esuli (2008). Our approach uses a Roget-like thesaurus, and it does not use any supervised classifiers.

Turney and Littman (2003) proposed a minimally supervised algorithm to calculate the semantic orientation of a word by determining if its tendency to co-occur with a small set of positive words is greater than its tendency to co-occur with a small set of negative words. They show that their approach performs better when it has a large amount of text at its disposal. They use text from 350 million web-pages (more than 100 billion words). Our approach does not make use of any text corpora, although co-occurrence statistics could be used to further improve the lexicon. Furthermore, our lexicon has entries for commonly used multi-word expressions as well.

Mohammad et al. (2008b) developed a method to determine the degree of antonymy (contrast) between two words using the *Macquarie The-*

[2] http://sentiwordnet.isti.cnr.it/

saurus (Bernard, 1986), co-occurrence statistics, and a small set of antonym-generating affix patterns such as X–*dis*X. Often, one member of a pair of contrasting terms is positive and one member is negative. In this paper, we describe how a subset of those affix patterns can be used in combination with a thesaurus and the edicts of marking theory to create a large lexicon of words and phrases marked with their semantic orientation.

3 Generating the Semantic Orientation Lexicon

Our algorithm to generate a semantic orientation lexicon has two steps: (1) identify a seed set of positive and negative words; (2) use a Roget-like thesaurus to mark the words synonymous with the positive seeds "positive" and words synonymous with the negative seeds "negative". The two steps are described in the subsections below. Our implementation of the algorithm used the *Macquarie Thesaurus* (Bernard, 1986). It has about 100,000 unique words and phrases.

3.1 Seed words

3.1.1 Automatically identifying seed words

It is known from marking theory that overtly marked words, such as *dishonest, unhappy,* and *impure*, tend to have negative semantic orientation, whereas their unmarked counterparts, *honest, happy,* and *pure*, tend to have positive semantic orientation (Lehrer, 1974; Battistella, 1990). Exceptions such as *biased–unbiased* and *partial–impartial* do exist, and in some contexts even a predominantly negative marked word may be positive. For example *irreverent* is negative in most contexts, but positive in the sentence below:

> *Millions of fans follow Moulder's irreverent quest for truth.*

However, as we will show through experiments, the exceptions are far outnumbered by those that abide by the predictions of marking theory.

We used a set of 11 antonym-generating affix patterns to generate overtly marked words and their unmarked counterparts (Table 1). Similar antonyms-generating affix patterns exist for many languages (Lyons, 1977). The 11 chosen affix patterns generated 2,692 pairs of marked and unmarked valid English words that are listed in the *Macquarie Thesaurus*. The marked words

Affix pattern		**# word**	
w_1	w_2	**pairs**	**example word pair**
X	*dis*X	382	*honest–dishonest*
X	*im*X	196	*possible–impossible*
X	*in*X	691	*consistent–inconsistent*
X	*mal*X	28	*adroit–maladroit*
X	*mis*X	146	*fortune–misfortune*
X	*non*X	73	*sense–nonsense*
X	*un*X	844	*happy–unhappy*
X	X*less*	208	*gut–gutless*
*l*X	*ill*X	25	*legal–illegal*
*r*X	*ir*X	48	*responsible–irresponsible*
X*less*	X*ful*	51	*harmless–harmful*
Total		2692	

Table 1: Eleven affix patterns used to generate the seed set of marked and unmarked words. Here 'X' stands for any sequence of letters common to both words w_1 and w_2.

are deemed negative and the unmarked ones positive, and these form our seed set of positive and negative words. We will refer to this set of orientation-marked words as the **affix seeds lexicon (ASL)**. Note that some words may have multiple marked counterparts, for example, *trust–trustless* and *trust–mistrust*. Thus, ASL has more negative words (2,652) than positive ones (2,379). Also, the X*less*–X*ful* pattern generates word pairs that are both overtly marked; words generated from X*less* are deemed negative and words generated from X*ful* are deemed positive.

It should be noted that the affix patterns used here are a subset of those used in Mohammad et al. (2008b) to generate antonym pairs. The affix patterns ignored are those that are not expected to generate pairs of words with opposite semantic orientation. For instance, the pattern *imX-exX* generates word pairs such as *import–export* and *implicit–explicit* that are antonymous, but do not have opposite semantic orientations.

3.1.2 Using manually annotated seed words

Since manual semantic orientation labels exist for some English words (the GI lexicon), we investigated their usefulness in further improving the coverage and correctness of the entries in our lexicon. We used the GI words as seeds in the same way as the words generated from the affix patterns were used (Section 3.1.1).

3.2 Generalizing from the seeds

A published thesaurus such as the *Roget's* or *Macquarie* has about 1,000 categories, each consisting of on average 120 words and commonly used

SO lexicon	Mode of creation	Resources used	# entries	# positives	# negatives
ASL	automatic	11 affix rules	5,031	2,379 (47.3%)	2,652 (52.7%)
GI	manual	human SO annotation	3,605	1,617 (44.9%)	1,988 (55.1%)
GI-subset	manual	human SO annotation	2,761	1,262 (45.7%)	1,499 (54.3%)
MSOL(ASL)	automatic	thesaurus, 11 affix rules	51,157	34,152 (66.8%)	17,005 (33.2%)
MSOL(GI)	automatic	GI, thesaurus	69,971	25,995 (37.2%)	43,976 (62.8%)
MSOL(ASL and GI)	automatic	GI, thesaurus, 11 affix rules	76,400	30,458 (39.9%)	45,942 (60.1%)
PSL	mostly manual	GI, other sources	6,450	2,298 (35.6%)	4,485 (64.4%)
SWN	automatic	human SO annotation, WordNet, ternary classifiers	56,200	47,806 (85.1%)	8,394 (14.9%)
TLL	automatic	100 billion word corpus, minimal human SO annotation	3,596	1,625 (45.2%)	1,971 (54.8%)

Table 2: Key details of semantic orientation (SO) lexicons. ASL = affix seeds lexicon, GI = General Inquirer, MSOL = Macquarie semantic orientation lexicon, PSL = Pitt subjectivity lexicon, SWN = SentiWordNet, TLL = Turney–Littman lexicon.

multi-word expressions. Terms within a category tend to be closely related, and they are further grouped into sets of near-synonymous words and phrases called **paragraphs**. There are about 10,000 paragraphs in most Roget-like thesauri.

Every thesaurus paragraph is examined to determine if it has a seed word (by looking up the seed lexicon described in Section 3.1). If a thesaurus paragraph has more positive seed words than negative seed words, then all the words (and multi-word expressions) in that paragraph are marked as positive. Otherwise, all its words are marked negative.

Note that this method assigns semantic orientation labels to word–**thesaurus paragraph** pairs. Thesaurus paragraphs can be thought of as word senses. A word with multiple meanings is listed in multiple thesaurus paragraphs, and so will be assigned semantic orientation labels for each of these paragraphs. Thus, the method assigns a semantic orientation to a word–sense combination similar to the SentiWordNet approach and differing from the General Inquirer and Turney–Littman lexicons.

However, in most natural language tasks, the intended sense of the target word is not explicitly marked. So we generate a word-based lexicon by asking the different senses of a word to vote. If a word is listed in multiple thesaurus paragraphs, then the semantic orientation label most common to them is chosen as the word's label. We will refer to this set of word–**semantic orientation** pairs as the **Macquarie Semantic Orientation Lexicon (MSOL)**. A set created from only the affix seeds will be called **MSOL(ASL)**, a set created from only the GI seeds will be called **MSOL(GI)**, and the set created using both affix seeds and GI seeds will be called **MSOL(ASL and GI)**.[3] We generated a similar word-based lexicon for SentiWordNet (SWN) by choosing the semantic orientation label most common to the synsets pertaining to a target word.

Table 2 summarizes the details of all the lexicons. MSOL(ASL and GI) has a much larger percentage of negatives than MSOL(ASL) because GI has a much larger percentage of negative words. These negative seeds generate many more negative entries in MSOL(ASL and GI). Of the 51,157 entries in MSOL(ASL), 47,514 are single-word entries and 3,643 are entries for multi-word expressions. Of the 69,971 entries in MSOL(GI), 45,197 are single-word entries and 24,774 are entries for common multi-word expressions. Of the 76,400 entries in MSOL(ASL and GI), 51,208 are single-word entries and 25,192 are entries for common multi-word expressions. In our evaluation, we used only the single-word entries to maintain a level playing field with other lexicons.

4 Evaluation

We evaluated the semantic orientation lexicons both intrinsically (by comparing their entries to the General Inquirer) and extrinsically (by using them in a phrase polarity annotation task).

4.1 Intrinsic: Comparison with GI

Similar to how Turney and Littman (2003) evaluated their lexicon (TLL), we determine if the semantic orientation labels in the automatically generated lexicons match the semantic orientation la-

[3] MSOL is publicly available at: www.umiacs.umd.edu/~saif/WebPages/ResearchInterests.html.

Lexicon	All	Positives	Negatives
MSOL(ASL)	74.3	84.2	65.9
SWN	60.1	86.5	37.9
TLL	**83.3**	83.8	82.8

Table 3: The percentage of GI-subset entries (all, only the positives, only the negatives) that match those of the automatically generated lexicons.

bels of words in GI. GI, MSOL(ASL), SWN, and TLL all have 2,761 words in common. We will call the corresponding 2,761 GI entries the **GI-subset**.

Table 3 shows the percentage of GI-subset entries that match those of the three automatically-generated lexicons (MSOL(ASL), SWN, and TLL). (The differences in percentages shown in the table are all statistically significant—$p < 0.001$.) We do not show results for MSOL(GI), MSOL(ASL and GI), and the Pittsburgh subjectivity lexicon (PSL) because these lexicons were created using GI entries. TLL most closely matches the GI-subset, and MSOL matches the GI-subset more closely than SWN with the GI-subset. However, the goal of this work is to produce a high-coverage semantic orientation lexicon and so we additionally evaluate the lexicons on the extrinsic task described below.

4.2 Extrinsic: Identifying phrase polarity

The MPQA corpus contains news articles manually annotated for opinions and private states.[4] Notably, it also has polarity annotations (positive/negative) at the phrase-level. We conducted an extrinsic evaluation of the manually-generated and automatically-generated lexicons by using them to determine the polarity of phrases in the MPQA version 1.1 collection of positive and negative phrases (1,726 positive and 4,485 negative).

We used a simple algorithm to determine the polarity of a phrase: (1) If any of the words in the target phrase is listed in the lexicon as having negative semantic orientation, then the phrase is marked negative. (2) If none of the words in the phrase is negative and if there is at least one positive word in the phrase, then it is marked positive. (3) In all other instances, the classifier refrains from assigning a tag. Indeed better accuracies in phrase semantic orientation annotation can be obtained by using supervised classifiers and more sophisticated context features (Choi and Cardie, 2008). However, our goal here is only to use this task as a testbed for evaluating different semantic orientation lexicons, and so we use the method described above to avoid other factors from influencing the results.

Table 4 shows the performance of the algorithm when using different lexicons. The performance when using lexicons that additionally make use of GI entries—MSOL(GI), MSOL(ASL and GI), PSL, and a combined GI-SWN lexicon—is shown lower down in the table. GI–SWN has entries from both GI and SWN. (For entries with opposing labels, the GI label was chosen since GI entries were created manually.) Observe that the best F-scores are obtained when using MSOL (in both categories—individual lexicons and combinations with GI). The values are significantly better than those attained by others ($p < 0.001$).

5 Discussion

The extrinsic evaluation shows that our thesaurus- and affix-based lexicon is significantly more accurate than SentiWordNet. Moreover, it has a much larger coverage than the GI and Pitt lexicons. Observe also that the affix seeds set, by itself, attains only a modest precision and a low recall. This is expected because it is generated by largely automatic means. However, the significantly higher MSOL performance suggests that the generalization step (described in Section 3.2) helps improve both precision and recall. Precision is improved because multiple seed words vote to decide the semantic orientation of a thesaurus paragraph. Recall improves simply because non-seed words in a paragraph are assigned the semantic orientation that is most prevalent among the seeds in the paragraph.

5.1 Support for the Polyanna Hypothesis

Boucher and Osgood's (1969) Polyanna Hypothesis states that people have a preference for using positive words and expressions as opposed to using negative words and expressions. Studies have shown that indeed speakers across languages use positive words much more frequently than negative words (Kelly, 2000). The distribution of positive and negative words in MSOL(ASL) further supports the Polyanna Hypothesis as it shows that even if we start with an equal number of positive and negative seed words, the expansion of the positive set through the thesaurus is much more pro-

[4] http://www.cs.pitt.edu/mpqa

SO lexicon	All phrases P	R	F	Only positives P	R	F	Only negatives P	R	F
Individual lexicons									
ASL	0.451	0.165	0.242	0.451	0.165	0.242	0.192	0.063	0.095
GI	0.797	0.323	0.459	0.871	0.417	0.564	0.763	0.288	0.419
MSOL(ASL)	0.623	0.474	**0.539**	0.631	0.525	0.573	0.623	0.458	**0.528**
SWN	0.541	0.408	0.465	0.745	0.624	**0.679**	0.452	0.328	0.380
TLL	0.769	0.298	0.430	0.761	0.352	0.482	0.775	0.279	0.411
Automatic lexicons + GI information									
MSOL(GI)	0.713	0.540	0.615	0.572	0.470	0.516	0.777	0.571	**0.658**
MSOL(ASL and GI)	0.710	0.546	**0.617**	0.577	0.481	0.525	0.771	0.574	**0.658**
PSL	0.823	0.422	0.558	0.860	0.487	0.622	0.810	0.399	0.535
GI-SWN	0.650	0.494	0.561	0.740	0.623	**0.677**	0.612	0.448	0.517

Table 4: Performance in phrase polarity tagging. P = precision, R = recall, F = balanced F-score. The best F-scores in each category are marked in bold.

nounced than the expansion of the negative set. (About 66.8% of MSOL(ASL) words are positive, whereas only 33.2% are negative.) This suggests that there are many more near-synonyms of positive words than near-synonyms of negative ones, and so there are many more forms for expressing positive sentiments than forms for expressing negative sentiment.

5.2 Limitations

Some of the errors in MSOL were due to non-antonymous instantiations of the affix patterns. For example, *immigrate* is not antonymous to *migrate*. Other errors occur because occasionally the words in the same thesaurus paragraph have differing semantic orientations. For example, one paragraph has the words *slender* and *slim* (which, many will agree, are positive) as well as the words *wiry* and *lanky* (which many will deem negative). Both these kinds of errors can be mitigated using a complementary source of information, such as co-occurrence with other known positive and negative words (the Turney–Littman method).

5.3 Future work

Theoretically, a much larger Turney–Littman lexicon can be created even though it may be computationally intensive when working with 100 billion words. However, MSOL and TLL are created from different sources of information—MSOL from overtly marked words and a thesaurus, and TLL from co-occurrence information. Therefore, a combination of the two approaches is expected to produce an even more accurate semantic orientation lexicon, even with a modest-sized corpus at its disposal. This is especially attractive for low resource languages. We are also developing methods to leverage the information in an English thesaurus to create semantic orientation lexicons for a low-resource language through the use of a bilingual lexicon and a translation disambiguation algorithm.

6 Visualizing the semantic orientation of thesaurus categories

In recent years, there have been substantial developments in the field of information visualization, and it is becoming increasingly clear that good visualizations can not only convey information quickly, but are also an important tool for gaining insight into an algorithm, detecting systematic errors, and understanding the task. In this section, we present some preliminary visualizations that are helping us understand our approach beyond the evaluations described above.

As discussed in Section 3.1.1, the affix seeds set connects the thesaurus words with opposite semantic orientation. Usually these pairs of words occur in different thesaurus categories, but this is not necessary. We can think of these connections as relationships of contrast in meaning and semantic orientation, not just between the two words but also between the two categories. To better aid our understanding of the automatically determined category relationships we visualized this network using the Fruchterman-Reingold force-directed graph layout algorithm (Fruchterman and Reingold, 1991) and the NodeXL network analysis tool (Smith et al., 2009) [5].

Our dataset consists of 812 categories from the *Macquarie Thesaurus* and 27,155 antonym edges between them. There can be an edge from a cat-

[5] Available from http://www.codeplex.com/NodeXL

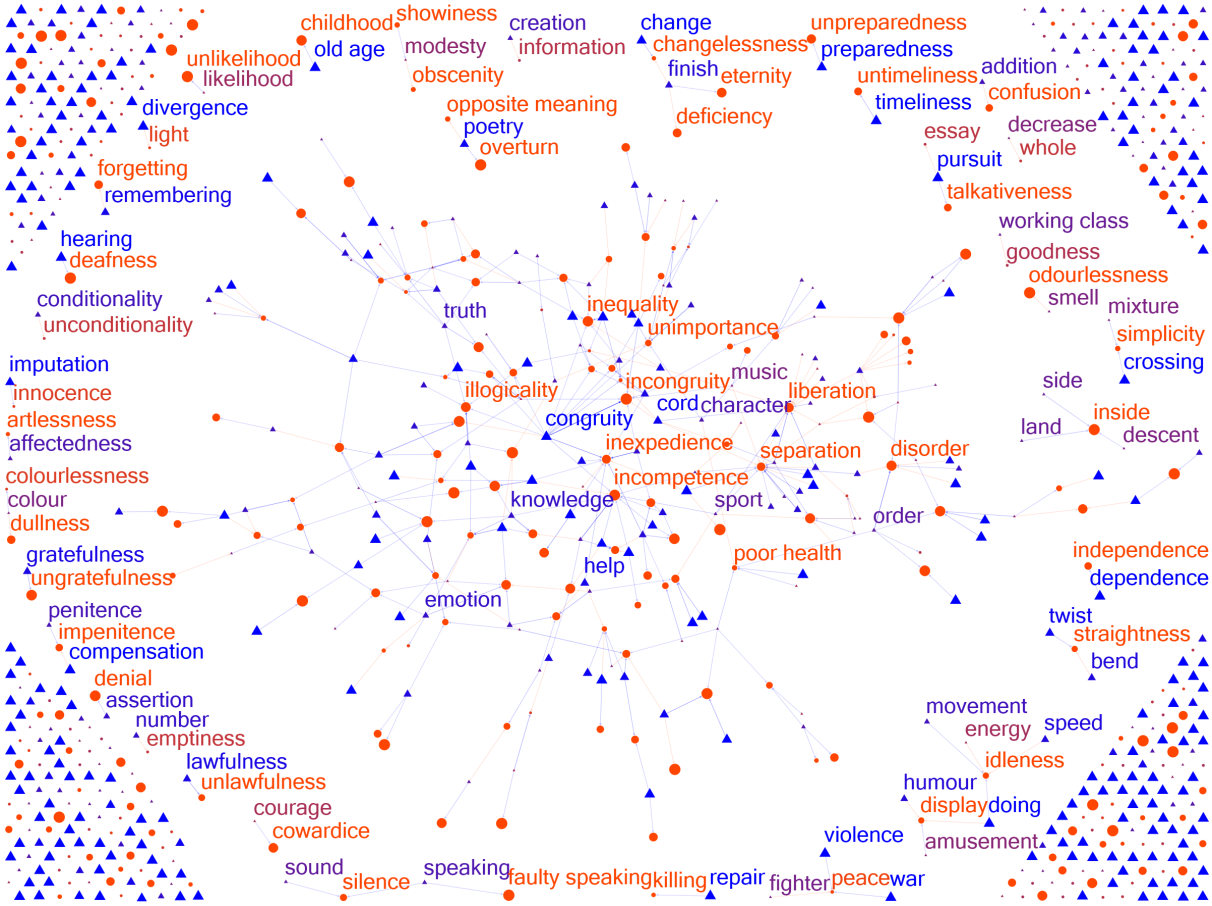


Figure 1: After removing edges with low weight we can see the structure the network backbone. Isolate category pairs are drawn in a ring around the main connected component and singletons are staggered in the corners. Each node is colored by its semantic orientation (red for negative, blue for positive) and edges are colored by their weight, from red to blue. Node shape also codes semantic orientation, with triangles positive and circles negative. Size codes the magnitude the semantic orientation, with the largest nodes representing the extremes. Node labels are shown for nodes in isolates and those in the top 20 for betweenness centrality.

egory to itself called a self-edge, indicating that a word and its antonym (with opposite semantic orientation) both exist in the same category. There can be multiple edges between two categories indicating that one or more words in one category have one or more antonyms in the other category. These multiple edges between category pairs were merged together resulting in 14,597 weighted meta-edges. For example, if there are n edges between a category pair they were replaced by a single meta-edge of weight n.

The network is too dense and interconnected for force-directed placement to generate a useful publication-size drawing of the entire network. By removing edges that had a weight less than 6, we can visualize a smaller and more understandable 540 edge network of the core categories and any new isolates created. Additionally, we show only edges between categories with opposite semantic orientations (Figure 1). Observe that there are three groups of nodes: those in the core connected component, the small isolates in the ring surrounding it, and the connectionless singletons arranged in the corners.

Each node c (thesaurus category) is colored on a red to blue continuous scale according to its semantic orientation SO, which is computed purely from its graph structure (in-degree ID and out-degree OD):

$$SO(c) = \frac{OD(c) - ID(c)}{OD(c) + ID(c)} \quad (1)$$

Blue nodes represent categories with many positive words; we will call them **positive cate-**

gories (p). Red nodes are categories with many negative words; we will call them **negative categories (n)**. Shades of purple in between are categories that have words with both positive and negative semantic orientation (**mixed categories**). Similarly, edges are colored according to their weight from red (small weight) to blue (large weight). We also use shape coding for semantic orientation, with triangles being positive and circles negative, and the size of the node depicts the magnitude of the semantic orientation. For example, the pair HEARING(p)–DEAFNESS(n) in the top left of Figure 1 represent the two size extremes: HEARING has a semantic orientation of 1 and DEAFNESS has a score of -1. The mixed categories with near 0 semantic orientation such as LIKELIHOOD with a score of .07 are the smallest.

Nodes are labeled by the thesaurus-provided head words—a word or phrase that best represents the coarse meaning of the category. For readability, we have restricted the labels to nodes in the isolates and the top 20 nodes in the core connected component that have the highest **betweenness centrality**, which means they occur on more shortest paths between other nodes in the network (i.e., they are the bridges or gatekeepers).

From the ring of isolates we can see how many antonymous categories, and their semantic orientations, are correctly recognized. For example, ASSERTION(p)–DENIAL(n), HEARING(p)–DEAFNESS(n), GRATEFULNESS(p)–UNGRATEFULNESS(n), and so on. Some codings may seem less intuitive, such as those in the core, but much of this is the effect of abstracting away the low weight edges, which may have more clearly identified the relationships.

An alternative approach to removing edges with low weight is to filter categories in the network based on graph-theoretic metrics like betweenness centrality, **closeness centrality**, and **eigenvector centrality**. We discussed betweenness centrality before. The closeness centrality of a node is the average distance along the shortest path between that node and all other nodes reachable from it. Eigenvector centrality is another measure of node importance, assigning node score based on the idea that connections to high-scoring nodes are more important than those to low-scoring ones. We removed nodes with less than 0.1 betweenness centrality, less than 0.04 eigenvector centrality, and above 2.1 closeness centrality, leaving the key 56 nodes. They have 497 edges between them, of which we show only those between categories with opposite semantic orientations (Figure 2). Node and edge color, size, and shape coding is as before.

Observe that most of these categories have a strongly evaluative nature. Also, as our algorithm makes connections using overt negative markers, it makes sense that the central categories in our network have negative orientation (negative categories have many words with overt markings). It is interesting, though, how some positive and mixed categories reside in the core too. Further inspection revealed that these categories have a large number of words within them. For example, it may be less intuitive as to why the category of MUSIC is listed in the core, but this is because it has about 1,200 words in it (on average, each category has about 120 words), and because many of these words, such as *harmonious(p), melodious(n),* and *lament(n)* are evaluative in nature.

7 Conclusion

We created a high-coverage semantic orientation lexicon using only affix rules and a Roget-like thesaurus. The method does not require terms with manually annotated semantic orientation labels, though we show that if available they can be used to further improve both the correctness of its entries and its coverage. The lexicon has about twenty times as many entries as in the General Inquirer and the Turney–Littman lexicons, and includes entries for both individual words and common multi-word expressions. Experiments show that it has significantly more correct entries than SentiWordNet. The approach is complementary to that of Turney and Littman (2003) and a combination of this approach with co-occurrence statistics (even if drawn from a modest sized corpus) is expected to yield an even better lexicon.

Visualization of the thesaurus categories as per the semantic orientations assigned to them by our algorithm reveals that affix patterns produce a strongly connected graph and that indeed there are many long chains of positive and negative categories. Furthermore, the key categories of this graph (the ones with high centrality and closeness) are strongly evaluative in nature, and most of them tend to have negative semantic orientation.

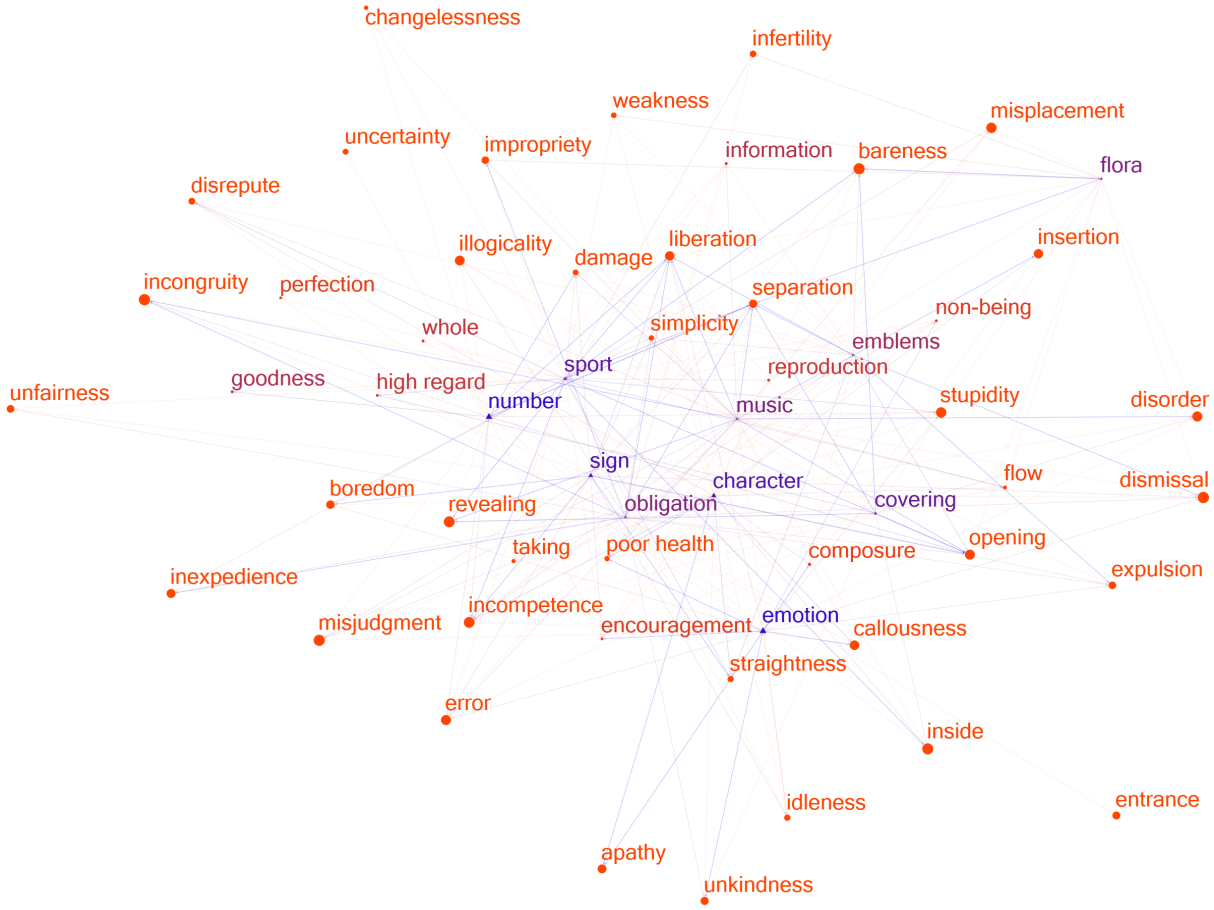


Figure 2: After filtering out nodes based on graph-theoretic metrics, the core of the network becomes visible. The visualization is colored as in Figure 1, and we can see how the core is dominated by categories with negative semantic orientation (red). Shape, size, and color coding is as before.

Acknowledgments

We thank Douglas W. Oard, Ben Schneiderman, Judith Klavans and the anonymous reviewers for their valuable feedback. This work was supported, in part, by the National Science Foundation under Grant No. IIS-0705832, in part, by the Human Language Technology Center of Excellence, and in part, by Microsoft Research for the NodeXL project. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsor.

References

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. *Text, Speech and Dialogue*, 4629:196–205.

Alina Andreevskaia and Sabine Bergler. 2006. Mining WordNet for fuzzy sentiment: Sentiment tag extraction from WordNet glosses. In *Proceedings of the EACL*, Trento, Italy.

Edwin Battistella. 1990. *Markedness: The Evaluative Superstructure of Language*. State University of New York Press, Albany, New York.

John R. L. Bernard, editor. 1986. *The Macquarie Thesaurus*. Macquarie Library, Sydney, Australia.

Jerry D. Boucher and Charles E. Osgood. 1969. The pollyanna hypothesis. *Journal of Verbal Learning and Verbal Behaviour*, 8:1–8.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP-2008)*, Waikiki, Hawaii.

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, pages 417–422, Genoa, Italy.

Andrea Esuli. 2008. *Automatic Generation of Lexical Resources for Opinion Mining: Models, Algorithms and Applications*. Ph.D. thesis, Department of Information Engineering, University of Pisa, Pisa, Italy.

Thomas M. J. Fruchterman and Edward M. Reingold. 1991. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164.

Gregory Grefenstette, Yan Qu, David Evans, and James Shanahan. 2004. Validating the coverage of lexical resources for affect analysis and automatically classifying new words along semantic axes. In James Shanahan Yan Qu and Janyce Wiebe, editors, *Exploring Attitude and Affect in Text: Theories and Applications, AAAI-2004 Spring Symposium Series*, pages 71–78, San Jose, California.

Vasileios Hatzivassiloglou and Kathleen McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of EACL*, pages 174–181, Madrid, Spain.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of ACM SIGKDD International ConferenceDiscovery and Data Mining (KDD-04)*, Seattle, WA.

Mario Jarmasz and Stan Szpakowicz. 2003. Roget's Thesaurus and semantic similarity. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-2003)*, pages 212–219, Borovets, Bulgaria.

Jaap Kamps, Maarten Marx, Robert J. Mokken, and Maarten de Rijke. 2004. Using WordNet to measure semantic orientation of adjectives. In *LREC*.

Hiroshi Kanayama and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 355–363, Sydney, Australia, July. Association for Computational Linguistics.

Michael H. Kelly. 2000. Naming on the bright side of life. volume 48, pages 3–26.

Adrienne Lehrer. 1974. *Semantic fields and lexical structure*. North-Holland; American Elsevier, Amsterdam and New York.

Lucian Vlad Lita, Andrew Hazen Schlaikjer, WeiChang Hong, and Eric Nyberg. 2005. Qualitative dimensions in question answering: Extending the definitional QA task. In *Proceedings of AAAI*, pages 1616–1617. Student abstract.

John Lyons. 1977. *Semantics*, volume 1. Cambridge University Press.

Saif Mohammad and Graeme Hirst. 2006. Distributional measures of concept-distance: A task-oriented evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2006)*, pages 35–43, Sydney, Australia.

Saif Mohammad, Bonnie Dorr, Melissa Egan, Jimmy Lin, and David Zajic. 2008a. Multiple alternative sentence compressions and word-pair antonymy for automatic text summarization and recognizing textual entailment. In *Proceedings of the Text Analysis Conference (TAC-2008)*, Gaithersburg, MD.

Saif Mohammad, Bonnie Dorr, and Graeme Hirst. 2008b. Computing word-pair antonymy. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Waikiki, Hawaii.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135.

Yohei Seki, Koji Eguchi, and Noriko Kando. 2004. Analysis of multi-document viewpoint summarization using multi-dimensional genres. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*, pages 142–145.

Marc Smith, Ben Shneiderman, Natasa Milic-Frayling, Eduarda Mendes Rodrigues, Vladimir Barash, Cody Dunne, Tony Capone, Adam Perer, and Eric Gleave. 2009. Analyzing (social media) networks with NodeXL. In *C&T '09: Proc. Fourth International Conference on Communities and Technologies*, Lecture Notes in Computer Science. Springer.

Swapna Somasundaran, Theresa Wilson, Janyce Wiebe, and Veselin Stoyanov. 2007. QA with attitude: Exploiting opinion type analysis for improving question answering in on-line discussions and the news. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*.

Philip Stone, Dexter Dunphy, Marshall Smith, and Daniel Ogilvie. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. MIT.

Carlo Strapparava and Alessandro Valitutti. 2004. WordNet-affect: and affective extension of WordNet. In *Proceedings of LREC*, Lisbon, Portugal.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientation of words using spin model. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 133–140.

Junichi Tatemura. 2000. Virtual reviewers for collaborative exploration of movie reviews. In *Proceedings of Intelligent User Interfaces (IUI)*, pages 272–275.

Loren Terveen, Will Hill, Brian Amento, David McDonald, and Josh Creter. 1997. PHOAKS: A system for sharing recommendations. *Communications of the Association for Computing Machinery (CACM)*, 40(3):59–62.

Peter Turney and Michael Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346.

Peter Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of ACL*, pages 417–424, Philadelphia, Pennsylvania.

Janyce M. Wiebe. 1994. Tracking point of view in narrative. *Computational Linguistics*, 20(2):233–287.

Theresa Wilson, Janyce Wiebe, and Paul Hoffman. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT-EMNLP*, pages 347–354, Vancouver, Canada.

Hong Yu and Vassileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of EMNLP*, pages 129–136, Morristown, NJ.

Matching Reviews to Objects using a Language Model

Nilesh Dalvi **Ravi Kumar** **Bo Pang** **Andrew Tomkins**
Yahoo! Research
701 First Ave
Sunnyvale, CA 94089
{ndalvi,ravikumar,bopang,atomkins}@yahoo-inc.com

Abstract

We develop a general method to match unstructured text reviews to a structured list of objects. For this, we propose a language model for generating reviews that incorporates a description of objects and a generic review language model. This mixture model gives us a principled method to find, given a review, the object most likely to be the topic of the review. Extensive experiments and analysis on reviews from Yelp show that our language model-based method vastly outperforms traditional tf-idf-based methods.

1 Introduction

Consider a user searching for reviews of "Casablanca Moroccan Restaurant." The search engine would like to obtain as many reviews of this restaurant as possible, both to offer a high-quality result set for even obscure restaurants, and to enable advanced applications such as aggregation/summarization/categorization of reviews and recommendation of alternate restaurants. To solve this problem, it faces two high-level challenges: first, identify the restaurant review pages on the Web; and second, given a review, identify the restaurant that is being reviewed. There has been previous work addressing the first challenge (Section 2). We focus in this paper on the second.

The Web is replete with restaurant reviews available on top restaurant verticals such as Yelp and CitySearch, as well as newspaper articles, newsgroup discussions, blog posts, small local review aggregators and so forth. Ideally, the search engine would like to obtain reviews from all possible sources. While identifying the subject matter of a given review on the large sites may be amenable to structured extraction through wrapper induction or related techniques, it is typically not cost-effective to apply such techniques to smaller "tail" sites, and purely unstructured sources require alternate approaches altogether. In this paper, we explore the setting of matching reviews to objects using only their textual content. Note that matching reviews to objects is a pervasive problem beyond the restaurant domain. Shopping verticals like to aggregate camera reviews, entertainment verticals wish to collect movie reviews, and so on. We use restaurant reviews as a running example, but the techniques are general.

More specifically, the problem we consider in this paper is the following. Given a list of structured objects (restaurants/cameras/movies) and a text review, identify the object from the list that is the topic of the review. Our focus on textual content allows us to expand the universe of sources from which we can extract reviews to include sources that are purely textual, such as forum posts, blog posts, newsgroup postings, and the like. In fact, even among collections of "structured" sources like review aggregators, there are no highly accurate unsupervised techniques to match a known review page to an object. Structured (e.g., HTML) cues provide valuable leverage in attacking this problem, but the types of textual cues we focus on are also a key part of the puzzle; in such a context, our techniques can still contribute to the overall matching problem.

It is important to contrast our problem against two settings of related flavor — *entity matching*, whose goal is to find the correspondence between two structured objects and *information retrieval (IR)*, whose goal is to match unstructured short text (query) against unstructured text (document).

Our problem is considerably harder than entity matching for the following reasons. In matching two structured objects there is often a natural correspondence between their attributes, whereas no such correspondence exists between an object and

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 609–618,
Singapore, 6-7 August 2009.

its review. For instance, while trying to match a review to a restaurant object, it is unclear if a specific portion of the review refers to the name of the restaurant, or to its location, or is a statement not concerning specifics of the restaurant. Moreover, even if we wish to use entity matching, we must first recognize the entities from a review. There are two methods to do this, namely, wrapper induction and information extraction. Wrapper induction methods have serious limitations: they are applicable only to highly-structured websites and involve human labeling effort that is expensive and error-prone and entails constant maintenance to keep wrappers up-to-date. Information extraction methods (Cardie, 1997; Sarawagi, 2008), on the other hand, often have limited accuracy.

Our problem is also not amenable to classical IR methods such as tf-idf. For example, suppose we want to find the relevant restaurant for a given review. The standard tf-idf will treat the review as the query, the set of restaurant as documents and compute the tf-idf scores. Now consider a restaurant called "Food."[1] Since the term "food" is rare as a restaurant name, it will get a very high idf score and hence will likely be the top match for all reviews containing the word "food." In fact, unlike in traditional IR, a "query" (i.e., review) is long and a "document" (i.e., restaurant) is short — this demands adapting established IR concepts such as inverse document frequency and document length normalization to our setting. If we take the opposite view by considering reviews as documents and restaurants as queries, we still deviate from the IR setting, since now we need to rank and find the best "query" for a given "document." In Section 3.4, we illustrate the shortcomings of both these approaches.

In fact, the nature of the object database we consider provides several unique opportunities over traditional IR. First the "document", i.e., the object to be matched, has more semantics, since each document is associated with one or more semantic attribute, such as the name/location of the restaurant. Second, the "query", i.e., the text we are matching is known to be a review of the object, and hence is rendered in a language that is "review-like" — this can be modeled by a generative process that produces reviews from objects. Third, the set of objects we are interested in is given a priori, and we only seek to match reviews with one of these objects; this makes our problem more tractable than open-ended entity recognition.

Our contributions. We propose a general method to match reviews to objects. To this end, we postulate a language model for generating reviews. The intuition behind our model is simple and natural: when a review is written about an object, each word in the review is drawn either from a description of the object or from a generic review language that is independent of the object. This mixture model leads to a method to find, given a review, the object most likely to be the topic of the review.

Our method is light-weight and scalable and can be viewed as obviating the need for highly-expensive information extraction. Since the method is text-based and does not rely on any HTML structural clues, it is especially applicable to reviews present in blogs and the so-called tail web sites — web sites for which it is not feasible to maintain wrappers to automatically extract the object of a review.

We then report results on over 11K restaurant reviews from Yelp. The experiments and our extensive analysis show that our language model-based method significantly outperforms traditional tf-idf based methods, which fail to take full advantage of the properties that are specific to our setting.

2 Related work

Opinion topic identification is the work closest to ours. In a recent paper, Stoyanov and Cardie (2008) approach this problem by treating it as an exercise in topic coreference resolution. Though they have to deal with topic ambiguities and a lack of explicit topic mentions as in our case, their notion of a topic is not driven by a structured listing. There has been some work on fine-grained opinion extraction from reviews (Kobayashi et al., 2004; Yi et al., 2003; Popescu and Etzioni, 2005; Hu and Liu, 2004); see (Pang and Lee, 2008) for a comprehensive survey. Most of this body of work focused on identifying product features of the object under review, rather than identifying the product itself. Note that while a dictionary of products is often more readily available than a dictionary of product features, identifying objects of reviews is non-trivial even with the help of the former. Indeed, it has been reported that lexicon-

[1] 1569 Lexington Ave., New York, NY 10029. (212) 348-0200.

lookup methods have limited success on general non-product review texts (Stoyanov and Cardie, 2008). In general, this line of work is more rooted in the information extraction literature, where text spans covering the object (or features of the object) were extracted as the first step; in contrast, we do not have an explicit extraction phase. Since the (very extensive) list of candidate objects are given as input, our task is to rank all matching objects, and in this sense is closer in nature to information retrieval tasks. There has been some work on detecting reviews in large-scale collections (Ng et al., 2006; Barbosa et al., 2009); this is a logical step that precedes the review matching step, the topic of our paper.

Language modeling is becoming a powerful paradigm in the realm of information retrieval applications (Ponte and Croft, 1998; Hiemstra, 1998; Song and Croft, 1999; Lafferty and Zhai, 2003; Zhai, 2008). The basic theme behind language modeling is to first postulate a model for each document and for a given query select the document that is most likely to have generated the query; smoothing is an important means to manage data sparsity in language models (Zhai and Lafferty, 2004). As noted earlier, language models developed for IR are unsuitable for our setting. Furthermore, there are opportunities, such as the presence of structure in our data, which we use in this work (Section 3.2). In fact, in a subsequent paper, we show how a language model specific to each attribute can further improve the accuracy of review matching (Dalvi et al., 2009).

Entity matching is a well-studied topic in databases. There are several approaches to entity matching: non-relational approaches, which consider pairwise attribute similarities between entities (Newcombe et al., 1959; Fellegi and Sunter, 1969), relational approaches, which exploit the relationships that exist between entities (Ananthakrishna et al., 2002; Kalashnikov et al., 2005), and collective approaches, which exploit the relationship between various matching decisions, (Bhattacharya and Getoor, 2007; McCallum and Wellner, 2004). The EROCS system (Chakaravarthy et al., 2006), which uses information extraction and entity matching, is closest in spirit to our problem; they, however, employ tf-idf to match, which we show to be significantly sub-optimal in our setting.

3 Model and method

In this section we present the problem formulation, the basic generative model for reviews, a method based on this model to associate an object with a review, and the techniques to estimate the parameters of this model.

Problem formulation. Let $\mathcal{E}$ denote a set of objects. Each object $e \in \mathcal{E}$ has a set of attributes and let $\text{text}(e)$ denote the union of the textual content of all its attributes. Suppose we have a collection of reviews $\mathcal{R}$, where each review is written (mainly) about one of the objects in the listing $\mathcal{E}$. The problem now is to correctly associate each $r \in \mathcal{R}$ with exactly one of $e \in \mathcal{E}$.

We model each review as a bag of words. Therefore, notation such as "$w \in r$" for a word w and a review r makes sense. For a review r and an object e, let $r_e = r \cap \text{text}(e)$.

As a running example, we use $\mathcal{E}$ to denote the set of all restaurants and $\mathcal{R}$ to denote the set of all restaurant reviews.

3.1 A generative model for reviews

We first state the intuition behind our generative model: when a review r is written about an object e, some words in r (e.g., the name and the address of the restaurant) are drawn from $\text{text}(e)$ to refer to the object under discussion, while some other words are drawn from a *generic review language* independent of e.

Formally, let $\alpha \in (0, 1)$ be a parameter. Let $P_e(\cdot)$ denote a distribution whose support is $\text{text}(e)$; this corresponds to the distribution of words specific to the object e, taken from the description $\text{text}(e)$. We use $P_e(w)$ to denote the probability the word w is chosen according to this distribution. Let $P(\cdot)$ be an object-independent distribution whose support is the review language, i.e., all the words that can be used to write a review; we use $P(w)$ to denote the probability the word w is chosen according to this distribution. Now, for a given object e, a review r is generated as follows. Each word in r is generated independently: with probability α, a word w is chosen with probability $P_e(w)$ and with probability $1 - \alpha$, a word w is chosen with probability $P(w)$. Thus, the review generation process is a multinomial, where the underlying process is a mixture of object-specific language and a generic review language.

Given a review r and an object e, by our independence assumption,

$$\begin{aligned}\Pr[r \mid e] &= Z(r) \prod_{w \in r} \Pr[w \mid e] \\ &= Z(r) \prod_{w \in r} ((1-\alpha)P(w) + \alpha P_e(w)), \quad (1)\end{aligned}$$

where $Z(r)$ is a normalizing term that only depends on the length of r and the counts of the words in it. Recalling $r_e = r \cap \text{text}(e)$, we note that $P_e(w)$ assigns zero probability to $w \notin r_e$. From (1), we get

$$\begin{aligned}\Pr[r \mid e] &= Z(r) \prod_{w \in r \setminus r_e} (1-\alpha)P(w) \cdot \\ &\quad \prod_{w \in r_e} ((1-\alpha)P(w) + \alpha P_e(w)) \\ &= Z(r) \prod_{w \in r} (1-\alpha)P(w) \cdot \\ &\quad \prod_{w \in r_e} \left(1 + \frac{\alpha}{1-\alpha} \frac{P_e(w)}{P(w)}\right). \quad (2)\end{aligned}$$

Note that Eq. (2) appears similar to the formula obtained in the language model approach for IR (Hiemstra and Kraaij, 1998); the interpretation of terms, however, is very different. For instance, $P(w)$ in our case is computed over the "query" corpus whereas the analogous term (collection frequency) in (Hiemstra and Kraaij, 1998) is computed over the "document" corpus. As the "Food" restaurant example in Section 1 suggests, using the "document" frequency is undesirable. The use of "query" corpus frequency arises naturally from our generative story and also guides us to a different way to estimate $P(w)$; see Section 3.3.

3.2 Matching a review to an object

Given the above review language model (RLM), we now state how to match a given review to an object. According to our model, the most likely object e^* to have generated a review r is given by

$$e^* = \arg\max_e \Pr[e \mid r] = \arg\max_e \frac{\Pr[e]}{\Pr[r]} \cdot \Pr[r \mid e].$$

In the absence of any information, we assume a uniform distribution for $\Pr[e]$. (Additional information about objects, such as their rating/popularity, can be used to model $\Pr[e]$ more accurately.) From this, we get

$$e^* = \arg\max_e \Pr[r \mid e],$$

or equivalently,

$$e^* = \arg\max_e \log \Pr[r \mid e].$$

Since $Z(r) \prod_{w \in r}((1-\alpha)P(w))$ is independent of e, using (2), we have

$$e^* = \arg\max_e \sum_{w \in r_e} \log \left(1 + \frac{\alpha}{1-\alpha} \frac{P_e(w)}{P(w)}\right). \quad (3)$$

3.3 Estimating the parameters

We now describe how to estimate the parameters of the model, namely, $P(\cdot)$, $P_e(\cdot)$, and α.

Recall that $P(\cdot)$ is the distribution of generic review language. Ideally, for each review r, if we know the component $r^{(e)}$ that came from the distribution $P_e(\cdot)$ and the component $r^{(g)}$ that came from $P(\cdot)$, then we can collect the $r^{(g)}$ components of all the reviews in $\mathcal{R}$, denoted as $\mathcal{R}^{(g)}$, and estimate $P(\cdot)$ by the fraction of occurrences of w in $\mathcal{R}^{(g)}$. More specifically, let $c(w, \mathcal{R}^{(g)})$ denote the number of times w occurs in $\mathcal{R}^{(g)}$. With add-one smoothing, we estimate

$$P(w) = \frac{c(w, \mathcal{R}^{(g)}) + 1}{\sum_{w'} c(w', \mathcal{R}^{(g)}) + |V|},$$

where $|V|$ is the vocabulary size.

In reality, we only have access to r and not to the components $r^{(e)}$ and $r^{(g)}$. If we have an aligned review corpus $\mathcal{R}'$, where for each review r, we know the true object e that generated it, we can closely approximate $r^{(e)}$ with r_e.[2] Let no-obj($\mathcal{R}'$) be the set of processed reviews where for each review-object pair (r, e), words in $\text{text}(e)$ are removed from r. By treating no-obj($\mathcal{R}'$) as an approximation of $\mathcal{R}^{(g)}$, we can compute $P(w)$ in the aforementioned manner. If we only have access to a review collection $\mathcal{R}'$ with no object alignment, there are other ways to effectively approximate $\mathcal{R}^{(g)}$; see Section 5.3 for more details.

Unlike $P(\cdot)$, we cannot learn an individual language model $P_e(\cdot)$ for each e, since we cannot expect to have training examples of reviews for each possible object e in the dataset. Thus, we need a simpler way to model $P_e(w)$. The most naive way would be to assume a uniform distribution, i.e., $P_e(w) = 1/|\text{text}(e)|$. However, each word

[2] There can be exceptions to this, e.g., review of a restaurant called "Tasty Bites" might use the word "tasty" from the review language, but not to refer to the restaurant. Nonetheless, we believe these will be rare exceptions and will not have significant effect in the estimation of $P(\cdot)$.

in $\text{text}(e)$ may not be generated with equal probability. In our running example, consider the case when $\text{text}(e)$ contains the full name of the restaurant, i.e., "Casablanca Moroccan Restaurant." A review for this restaurant is more likely to choose the word "Casablanca" than any other word to refer to this restaurant since this is arguably more informative than "Moroccan" or "Restaurant." This can be captured by using the frequency f_w of the word w in $\mathcal{R}$ or in $\{\text{text}(e) \mid e \in \mathcal{E}\}$. For a suitable function $g(w)$ that is inversely growing as f_w (say, $g(w) = \log(1/f_w)$), we let

$$P_e(w) = \frac{g(w)}{\sum_{w' \in \text{text}(e)} g(w')}.$$

Alternatively, it is possible to construct models where $P_e(w)$ is more directly estimated from the data; in fact, one can also use suitable translation models to estimate $P_e(w)$ for w that may not even occur in $\text{text}(e)$ — this will help in cases where reviews use an abbreviation such as "Casa" or "CMR" to refer to our running example. Such models require either fine-grained labeled examples or, as we show in (Dalvi et al., 2009), more sophisticated estimation techniques.

It is tempting to assume that common words such as "Restaurant" may not contribute towards matching a review to an object and hence one can conveniently set $P_e(w) = 0$ for such words w. (Such a list of words can easily be compiled using a domain-specific stopword list.) This may hurt — in our example, the presence of the word "Restaurant" in a review might help to disambiguate the object of reference, if the listing were also to contain a "Casablanca Moroccan Cafe".

3.4 Properties of the model

Eq. (3) indicates that our method (denoted as RLM) gives less importance to common words with high $P(w)$. This corresponds to the intuition behind the standard tf-idf scheme. Why, then, do we expect RLM to be more effective? Here, we discuss the salient features of our method, contrasting it with tf-idf in particular.

First, we take a closer look at different ways to apply tf-idf techniques to our setting. Since the task is to find the most relevant object given a review, a naive way to apply the standard tf-idf (denoted TFIDF) will treat each review to be the query and each object to be a document and score documents using the standard tf-idf scoring. This, however, leads to severe problems since this computes the inverse document scores over the object corpus — recall the "Food" example in Section 1.

A more reasonable way to apply tf-idf is to instead treat objects as queries and reviews as documents for computing tf-idf scores (denoted TFIDF$^+$). For a word w, let $Q(w) = \frac{\text{df}(w)}{N}$, where N is the number of reviews in the corpus and $\text{df}(w)$ is the number of reviews containing w. Given a review r and an object e, the score of the object is given by $\sum_{w \in r_e} \log(1/Q(w))$, and we want to pick the object with the maximum score. As we will discuss later, document-length normalization (i.e., normalizing by object description length so that a restaurant with a long name does not get an unfair disadvantage) is still non-trivial here.

As noted earlier, Eq. (3), used by RLM for matching reviews with objects, has a striking resemblance to the TFIDF$^+$ scoring function. Both have the form

$$e^* = \arg\max_e \sum_{w \in r_e} \log f(w),$$

where for RLM,

$$f(w) = f_R(w) = 1 + \frac{\alpha}{1-\alpha}\frac{P_e(w)}{P(w)},$$

and for TFIDF$^+$,

$$f(w) = f_B(w) = \frac{1}{Q(w)}.$$

In both cases, $f(w)$ is monotonically decreasing in the frequency of w in the corpus. However, there are several differences between the two cases. We highlight some of them here, with the aim of illustrating the power of our review language model (RLM).

Object length normalization. First note that the $P_e(w)$ term in $f_R(w)$ acts as an object length normalizing term, i.e., it adds up to one for each e and weighs down $P(w)$ for objects with long $\text{text}(e)$. This also has the effect of penalizing reviews that are missing critical words in the object description. In contrast, $f_B(w)$ is unnormalized with respect to the object length. The standard document normalization techniques in IR do not apply well to our setting since our "documents" (i.e., object descriptions) are short. E.g., if the object description contains only one token, the standard cosine-normalization technique (Salton et al.,

1975) will yield a normalized score of 1 *irrespective* of the token. Thus for a review containing the words "Food" and "Casablanca", the standard normalization will yield the same score for a restaurant named "Food" and a restaurant named "Casablanca", ignoring the fact that "Food" is much more likely to be an object-independent term. Note that this only becomes a problem when the entire "document" is part of the match, which rarely happens in an IR setting where the documents are typically much longer than the queries. Indeed, in our experiments, we observe lower performance when we apply cosine-normalization to the tf-idf scores. On the other hand, in $f_R(w)$, the $P(w)$ term can still distinguish the two aforementioned objects even when $P_e(w)$ are equal.

Dampening. With $\alpha < 1$, $f_R(w)$ is effectively a dampened version of $\frac{P_e(w)}{P(w)}$. In other words, differences between very frequent words and very infrequent words are somewhat smoothed out. Indeed, if we modify TFIDF$^+$ by introducing a similar dampening factor into $f_B(w)$, we observe improvement in its performance (Section 5.4).

Removing mentions of an object. Another difference is that in RLM, $P(w)$ is estimated on reviews with object mentions removed, since the model indicate that $P(w)$ accounts for object-independent review language. In contrast, TFIDF$^+$ computes $Q(w)$ on full reviews. We illustrate the difference on the following example. Consider a review that reads "...Maggiano's has great Fondue." If "Maggiano's" and "Fondue" both occur the same number of times in the corpus, then they get the same idf (i.e., $Q(w)$) score. In RLM, however, "Maggiano's" will get much smaller probability in the generic review distribution $P(\cdot)$ than "Fondue", since "Maggiano's" almost always occurs in reviews as restaurant name mentions, thus is removed from the estimation of its $P(\cdot)$ probability. On the other hand, the word "Fondue" is more likely to retain higher probability in $P(\cdot)$ since it tends to appear as dish names. As a result, our model will assign higher weight to "Maggiano's Restaurant" than "Fondue Restaurant". As we can see, RLM evaluates the ability of a word to identify the review object rather than rely on the absolute rarity of the word, which is done by tf-idf.

Using term counts. One last difference is that $f_R(w)$ uses term counts of words rather than the standard document counts used by $f_B(w)$. Our evaluation suggests that at least in practice, this does not have a big impact on the overall accuracy.

In the experiments we show that these factors together account for the performance difference between RLM and tf-idf. Our model gives a principled way to introduce these factors, however.

4 Data

In this section we describe the dataset constructed for the task of matching restaurant reviews to the corresponding restaurant objects. Our goal is to obtain a large collection of reviews on which to estimate the generic language model, with a significant portion of them aligned with the objects for which the reviews were written; this portion will serve as the gold-standard test set.

To this end, we obtained a set of reviews from the Yelp website, `yelp.com`. This website contains a collection of reviews about various businesses and for each business, has a webpage containing the business information and a list of reviews. We crawled all restaurant pages from Yelp. For each restaurant, we extracted its name and city location from the business information section via HTML cues, and a list of no more than 40 reviews. We obtained the textual content of 299,762 reviews, each aligned with one of a set of 12,408 unique restaurants hosted on Yelp. Note that while our technique is not targeted for head sites like Yelp (where wrapper induction might be a more accurate approach), this provides a large-scale dataset, conveniently labeled with object information, and simulates the tail-site scenario where we rely heavily on the textual content of reviews to identify objects.

Many of the reviews in Yelp do not contain any identifying information. In fact, some of them are as short as "Great place. Awesome food!!". We processed the dataset to retain only reviews that mention the name of the restaurant, even if partially, and, when the restaurant name is a common word, also the city of the restaurant. Each of the remaining reviews is expected to have enough information for a human to identify the restaurant corresponding to the review.

To further increase the difficulty of the matching task, we obtained a much more extensive list of restaurant objects in the Yahoo! Local database, which contains 681,320 restaurants. Our task is to match a given Yelp review, using only its free-form textual content, with its corresponding

restaurant in the Yahoo! Local database. We then proceeded to generate the gold standard that contains the correct restaurant in the Yahoo! Local database for each review. We employed geocoding to match addresses across the two databases along with approximate name matches. Note that in the final dataset, only half of the restaurants have the exact same name listed in both Yelp and Yahoo! Local; this limits the success of naive dictionary-based methods.

The final aligned dataset contained 24,910 Yelp reviews ($\mathcal{R}$), covering 6,010 restaurants. We set aside half of the reviews ($\mathcal{R}'$) to estimate the models and the other half ($\mathcal{R}_{\text{test}}$) to evaluate our technique. We also set aside 1,000 reviews as development set, on which we conducted initial experiments. The total size of the test corpus, $\mathcal{R}_{\text{test}}$ was 11,217. The splitting of $\mathcal{R}$ into $\mathcal{R}'$, $\mathcal{R}_{\text{test}}$, and the development set was done in such a way that there are no overlapping restaurants between them. Also, the reviews that were filtered out because of lack of identifying information were added back to $\mathcal{R}'$ for learning the review language model, expanding $\mathcal{R}'$ to a total of 205,447 reviews.

5 Evaluation

In this section we evaluate our proposed review-language based matching algorithm RLM.

5.1 Experimental considerations

Baseline system. We use the TFIDF and TFIDF$^+$ algorithms described in Section 3.4 as baseline algorithms. Since we are comparing objects that can have varying lengths, we tried the standard cosine-normalization techniques for document length normalization. For reasons described in Section 3.4, however, the normalization significantly lowered the accuracy. All the numbers reported here are using tf-idf scores without normalization.

Efficiency. For both RLM and the baseline algorithms, it is impractical to compute the similarity of a review with each object in the database. Since all objects that do not intersect with the review have a zero score, we built an inverted index to retrieve all objects containing a given word. Even a simple inverted index can be very inefficient since for each review, words such as "Restaurant" or "Cafe" retrieve a substantial fraction of the whole database. Hence, we further optimized the index by looking at the document frequencies of the words and considering word bigrams in object descriptions. The index only retrieves objects that have a non-trivial overlap with the review; e.g., an overlap of "Casablanca" is considered non-trivial while an overlap of "Restaurant" is considered trivial. Once these candidates are retrieved, our scoring function takes into account all overlapping tokens.

For the YELP dataset, the index returns an average of 200 restaurants for each review. This points to the general difficulty of review matching over a large corpus of objects, since a simple dictionary-based named-entity recognition will hit at least 200 objects for many reviews.

Experiment settings. For RLM, we conducted initial experiments and performed parameter estimation on the development data. The experimental settings we used for RLM are as follows: we set $g(w) = \log(1/f_w)$ for P_e, where f_w is estimated on the review collection. $P(w)$ is estimated on all reviews in R', where for each review, all tokens of its corresponding $\text{text}(e)$, if present, are removed, in order to approximate the *generic* review language independent of e, as required by our generative model. We estimate α to be 0.002, tuned on the development set; in our experiments, we observe that the performance is not very sensitive to α.

5.2 Main results

In this section we present the main comparisons between RLM and the baseline in details.

Performance measure. Our task resembles a standard IR task in that our algorithm ranks candidate objects for a given review by their "aboutness" level. Unlike a standard IR task, however, we are not interested in retrieving multiple "relevant" objects, as each review in our dataset has only one single correct match from $\mathcal{E}$. A review match is correct if the top-1 prediction (i.e., e^*) is accurate. In what follows, we report the average accuracy for various experimental settings. Note that we can take the average accuracy over all reviews (reported as micro-average), regardless of which restaurants they are about; or we can first compute the average for reviews about the same restaurant, and report the average over all restaurants (macro-average). When not specified, we report the micro-average.

Main comparisons. Table 1(a) summarizes the main comparison. Our proposed algorithm RLM

Method	Micro-avg.	Macro-avg.
RLM	0.647	0.576
TFIDF$^+$	0.518	0.481
TFIDF	0.314	0.317

(a) Main comparison.

Method	Micro-avg.	Macro-avg.
RLM-UNIFORM	0.634	0.562
RLM-UNCUT	0.627	0.546
RLM-DECAP	0.640	0.573

(b) RLM variants.

Method	Micro-avg.	Macro-avg.
TFIDF$^+$-N	0.586	0.523
TFIDF$^+$-D	0.593	0.533
TFIDF$^+$-O	0.522	0.488
TFIDF$^+$-ND	0.628	0.549
TFIDF$^+$-NDO	0.647	0.576

(c) TFIDF$^+$ variants.

Table 1: Average accuracy of the top-1 prediction for various techniques. Micro-average computed over 11,217 reviews in $\mathcal{R}_{\text{test}}$; macro-average computed over 2,810 unique restaurants in $\mathcal{R}_{\text{test}}$.

clearly outperforms the TFIDF$^+$ baseline measured by either micro- or macro-average accuracy. The standard TFIDF, as predicted, performs the worst.

Some reviews can be particularly difficult to match, which can be reflected in a low matching score. Nonetheless, we predict the most likely object. Suppose we impose a threshold and return the most likely object only when its score is above threshold, we can then compute precision and recall at different thresholds. Figure 1 presents the precision–recall curve (using micro-average) for both RLM and TFIDF$^+$. Again, RLM clearly outperforms TFIDF$^+$ across the board.

We then generalize the definition of accuracy into accuracy@k: a review is considered as correctly matched if one of the top-k objects returned is the correct match. We plot accuracy@k as a function of k. While the gap between RLM and TFIDF$^+$ is smaller as k increases, RLM clearly outperforms TFIDF$^+$ for all $k \in \{1, \ldots, 10\}$.

One final comparison is accuracy@1 as a function of the review length. Given our current setting, longer reviews might be more difficult to match since they may include more proper nouns such as dish names and related restaurants, and

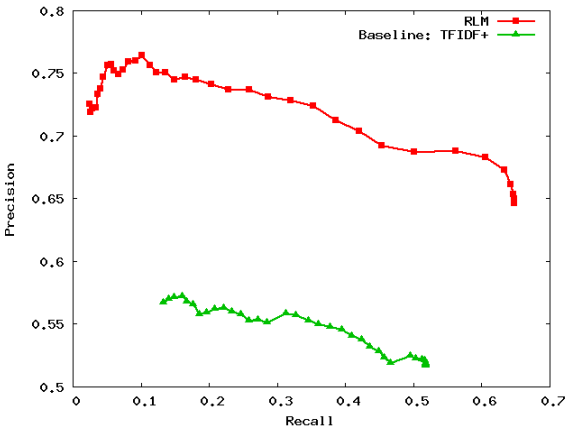


Figure 1: Precision–recall curve (of top one prediction): RLM vs. TFIDF$^+$ baseline.

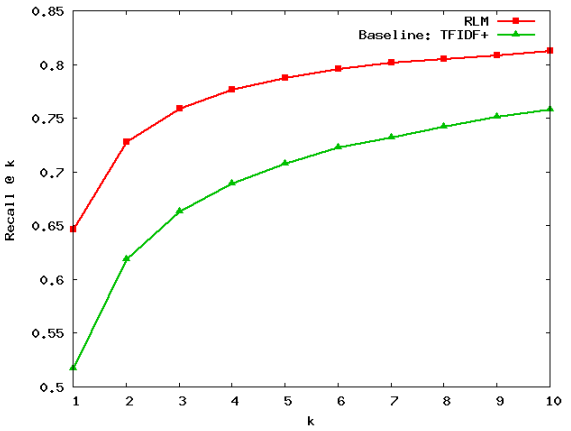


Figure 2: Accuracy@k (percentage of reviews whose correct match is returned in one of its top-k predictions): RLM vs. TFIDF$^+$ baseline.

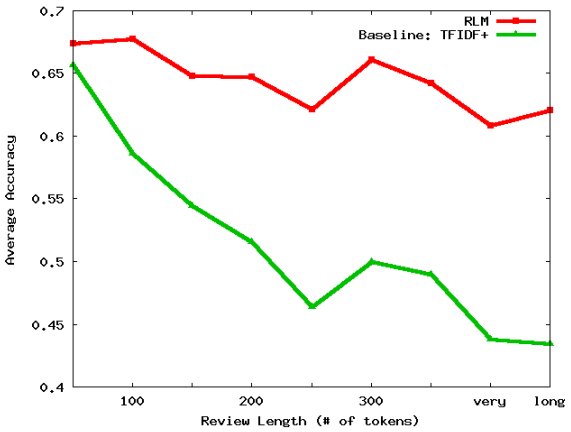


Figure 3: Average accuracy of the top-1 prediction for reviews with different length (on test set): RLM vs. TFIDF$^+$ baseline.

yield a longer list of highly competitive candidate objects. Interestingly, the gap between RLM and TFIDF$^+$ is much smaller for shorter reviews. As reviews get longer, the performance of RLM is relatively stable, whereas the performance of TFIDF$^+$ drops down significantly.

5.3 Experimental choices for RLM

We now examine the experimental choices we made for different components of RLM by defining the following variations of RLM.

RLM-UNIFORM: rather than setting $g(w) = \log(1/f_w)$ for P_e, we use the uniform distribution $P_e(w) = 1/|\text{text}(e)|$. From the third line of Table 1 (b), there is a slight accuracy drop of $\sim 1.3\%$.

RLM-UNCUT: suppose we only have access to a review corpus with no alignment to $\text{text}(e)$, and thus have to approximate $P(w)$ by estimating it on the set of original "un-cut" reviews, how much does that affect our performance? As indicated in the fourth row of Table 1 (b), this reduces accuracy by about 2% on our test data.

RLM-DECAP: as an alternative way to deal with lack of aligned data, we consider a variation of the above algorithm by removing all the capitalized words from un-annotated reviews. Clearly, this can result in both "over-cutting" and "under-cutting" of true restaurant name mentions. However, as indicated in the fourth row of Table 1 (b), this is very close to the best accuracy achieved. Thus, an effective model can be learned even without aligned data.

5.4 Revisiting TFIDF$^+$: what's amiss?

In this section we revisit the main differences between our model and the TFIDF$^+$ outlined in Section 3.4, and investigate their empirical importance by introducing these features into TFIDF$^+$ and examine their effectiveness in that framework.

Object length normalization. We consider a modified TFIDF$^+$ measure $f_M(w) = P_e(w)/Q(w)$, which we call TFIDF$^+$-N (normalized). As shown in Table 1 (c), this change alone can increase the average accuracy by nearly 7%.

Dampening. We consider a modified TFIDF$^+$ measure $f_M(w) = 1 + \beta \cdot \frac{N}{\text{df}(w)}$, which we call TFIDF$^+$-D. Table 1 (c) reports the performance of using this measure, with $\beta = 0.1$ (set on development data). Again, this measure alone can induce over 7% increase in accuracy. Indeed, combining normalization and dampening, (i.e., $f_M(w) = 1 + \beta \cdot P_e(w) \cdot \frac{N}{\text{df}(w)}$), denoted as TFIDF$^+$-ND, we get comparable performance to RLM-UNCUT.

Removing mentions of objects. Again, we can incorporate this in a heuristic way in TFIDF$^+$, which we denote by TFIDF$^+$-O. Interestingly, while using the original $f_B(w)$ function with $\text{df}(w)$ computed on the object-removed review collection does not yield a big improvement, this does bring the performance of the fully modified TFIDF$^+$ to the same level of the standard RLM (see line marked TFIDF$^+$-NDO.)

Using term counts. Our investigation suggests that at least in practice, using $Q(w)$ vs. $P(w)$ is not a critical decision, as a fully modified TFIDF$^+$ can achieve the same performance using $\text{df}(w)$ to quantify frequency of the word. Our experiments on this dataset show that each of the other modeling decisions incorporated in RLM is important.

6 Conclusions

We proposed a generative model for reviews where reviews are generated from the mixture of a distribution involving object terms and a generic review language model. The model provides us a principled way to match reviews to objects. Our evaluation on a real-world dataset shows that our techniques vastly outperforms standard tf-idf based techniques.

Acknowledgments

We thank Don Metzler for many discussions and the anonymous reviewers for their comments.

References

R. Ananthakrishna, S. Chaudhuri, and V. Ganti. 2002. Eliminating fuzzy duplicates in data warehouses. In *Proc. 28th VLDB*, pages 586–596.

L. Barbosa, R. Kumar, B. Pang, and A. Tomkins. 2009. For a few dollars less: Identifying review pages sans human labels. In *Proc. NAACL*.

I. Bhattacharya and L. Getoor. 2007. Collective entity resolution in relational data. *ACM TKDD*, 1(1).

C. Cardie. 1997. Empirical methods in information extraction. *AI Magazine*, 18(4):65–80.

V. T. Chakaravarthy, H. Gupta, P. Roy, and M. Mohania. 2006. Efficiently linking text documents with relevant structured information. In *Proc. 32nd VLDB*, pages 667–678.

N. Dalvi, R. Kumar, B. Pang, and A. Tomkins. 2009. A translation model for matching reviews to objects. Manuscript.

I. P. Fellegi and A. B. Sunter. 1969. A theory for record linkage. *JASIS*, 64:1183–1210.

D. Hiemstra and W. Kraaij. 1998. Twenty-one at TREC7: Ad-hoc and cross-language track. In *Proc. 7th TREC*, pages 174–185.

D. Hiemstra. 1998. A linguistically motivated probabilistic model of information retrieval. In *Proc. ECDL*, pages 569–584.

M. Hu and B. Liu. 2004. Mining opinion features in customer reviews. In *Proc. AAAI*, pages 755–760.

D. V. Kalashnikov, S. Mehrotra, and Z. Chen. 2005. Exploiting relationships for domain-independent data cleaning. In *Proc. 5th SDM*.

N. Kobayashi, K. Inui, Y. Matsumoto, K. Tateishi, and T. Fukushima. 2004. Collecting evaluative expressions for opinion extraction. In *Proc. 1st IJCNLP*, pages 596–605.

J. Lafferty and C. Zhai. 2003. Probabilistic relevance models based on document and query generation. In W. B. Croft and J. Lafferty, editors, *Language Modeling and Information Retrieval*. Academic Publishers.

A. McCallum and B. Wellner. 2004. Conditional models of identity uncertainty with application to noun coreference. In *Proc. 17th NIPS*.

H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. 1959. Automatic linkage of vital records. *Science*, 130:954–959.

V. Ng, S. Dasgupta, and S. M. Niaz Arifin. 2006. Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proc. 21st COLING/44th ACL*, pages 611–618.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

J. M. Ponte and W. B. Croft. 1998. A language modeling approach to information retrieval. In *Proc. 21st SIGIR*, pages 275–281.

A.-M. Popescu and O. Etzioni. 2005. Extracting product features and opinions from reviews. In *Proc. HLT/EMNLP*.

G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

S. Sarawagi. 2008. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377.

F. Song and W. B. Croft. 1999. A general language model for information retrieval. In *Proc. 22nd SIGIR*, pages 279–280.

V. Stoyanov and C. Cardie. 2008. Topic identification for fine-grained opinion analysis. In *Proc. COLING*.

J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack. 2003. Sentiment analyzer: Extrating sentiments about a given topic. In *Proc. 3rd ICDM*, pages 427–434.

C. Zhai and J. Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM TOIS*, 22(2):179–214.

C. Zhai. 2008. Statistical language models for information retrieval a critical review. *Foundations and Trends in Information Retrieval*, 2(3):137–213.

EEG responds to conceptual stimuli and corpus semantics

Brian Murphy
CIMeC, University of Trento
Rovereto 38068, Italy
brian.murphy@unitn.it

Marco Baroni
CIMeC, University of Trento
Rovereto 38068, Italy
marco.baroni@unitn.it

Massimo Poesio
CIMeC, University of Trento
Rovereto 38068, Italy
massimo.poesio@unitn.it

Abstract

Mitchell et al. (2008) demonstrated that corpus-extracted models of semantic knowledge can predict neural activation patterns recorded using fMRI. This could be a very powerful technique for evaluating conceptual models extracted from corpora; however, fMRI is expensive and imposes strong constraints on data collection. Following on experiments that demonstrated that EEG activation patterns encode enough information to discriminate broad conceptual categories, we show that corpus-based semantic representations can predict EEG activation patterns with significant accuracy, and we evaluate the relative performance of different corpus-models on this task.

1 Introduction

Models of semantic relatedness induced from corpus data have proven effective in a number of empirical tasks (Sahlgren, 2006) and there is increasing interest in whether distributional information extracted from corpora correlates with aspects of speakers' semantic knowledge: see Lund and Burgess (1996), Landauer and Dumais (1997), Almuhareb (2006), Padó and Lapata (2007), Schulte im Walde (2008), among many others. For this purpose, corpus models have been tested on datasets that are based on semantic judgements (metalinguistic or meta-cognitive intuitions about synonymy, semantic distance, category-membership) or behavioural experiments (semantic priming, property generation, free association). While all these data are valuable, they are indirect reflections of semantic knowledge, and when the predictions they make diverge from those of corpora, interpretation is problematic: is the corpus model missing essential aspects of semantics, or are non-semantic factors biasing the data elicited from informants?

Reading semantic processes and representations directly from the brain would be an ideal way to get around these limitations. Until recently, analysis of linguistic quantities using neural data collected with EEG (measurement at the scalp of voltages induced by neuronal firing) or fMRI (measurement of changes of oxygen concentrations in the brain tied to cognitive processes) had neither the advantages of corpora (scale) nor of informants (finer grained judgements).

However, some clear patterns of differential activity have been found for broad semantic classes. Viewing images of natural (typically animals and plants) and non-natural (typically artefacts like tools or vehicles) objects elicits different loci of activity in fMRI (Martin and Chao, 2001) and EEG (Kiefer, 2001), that persist across participants. Differences have also been found in response to auditorily or visually presented words of different lexical classes, such as abstract/concrete, and verb/noun (Pulvermüller, 2002). But interpretation of such group results remains somewhat difficult, as they may be consistent with more than one distinction: the natural/artefactual division just mentioned, may rather be between living/non-living entities, dynamic/static entities, or be based on embodied experience (e.g. manipulable or not).

More recently, however, machine learning and other numerical techniques have been successfully applied to extract semantic information from neural data in a more discriminative fashion, down to the level of individual concepts. The work presented here builds on two strands of previous work: Murphy et al. (2008) use EEG data to perform semantic categorisation on single stimuli; and Mitchell et al. (2008) introduce an fMRI-based method that detects word level distinctions by learning associations between features of neural activity and semantic features derived from a

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 619–627,
Singapore, 6-7 August 2009.

corpus. We combine these innovations by introducing a method that extracts featural representations from the EEG signal, and uses corpus-based models to predict word level distinctions in patterns of EEG activity. The proposed method achieves a performance level significantly above chance (also when distinguishing between concepts from the same semantic category, e.g., *dog* and *cat*), and approaching that achieved with fMRI.

The paper proceeds as follows. The next section describes a simple behavioural experiment where Italian-speaking participants had to name photographic images of mammals and tools while their EEG activity was being recorded, and continues to detail how the rich and multidimensional signals collected were reduced to a small set of optimally informative features using a new method. Section 3 describes a series of corpus-based semantic models derived from both a raw-text web corpus, and from various parsings of a conventional corpus. In Section 4 we describe the training of a series of linear models, that each learn the associations between a set of corpus semantic features and an individual EEG activity feature. By combining these models it is possible to predict the EEG activity pattern for a single unseen word, and compare this to the observed pattern for the corresponding concept. Results (Section 5) show that these predictions succeed at a level significantly above chance, both for coarser distinctions between words in different superordinate categories (e.g., differentiating between *drill* and *gorilla*), and, at least for the model based on the larger web corpus, for those within the same category (e.g., *drill* vs *spanner*, *koala* vs *gorilla*).

2 Neural Activation Data

2.1 Data collection

EEG data was gathered from native speakers of Italian during a simple behavioural experiment at the CIMeC/DiSCoF laboratories at Trento University. Seven participants (five male and two female; age range 25-33; all with college education) performed a silent naming task. Each of them was presented[1] on screen with a series of contrast-normalised greyscale photographs of tools (garden and work tools) and land mammals (excluding emotionally valent domesticated animals and predators), for which they had to think of the most appropriate name (see figure 1). They were not explicitly asked to group the entities into superordinate categories, or to concentrate on their semantic properties, but completing the task involved resolving each picture to its corresponding concept. Images remained on screen until a keyboard response was received from the participant to indicate a suitable label had been found, and presentations were interleaved with three second rest periods. Thirty stimuli in each of the two classes were each presented six times, in random order, to give a total of 360 image presentations in the session. Response rates were over 95%, and a post-session questionnaire determined that participants agreed on image labels in approximately 90% of cases. English terms for the concepts used are listed below.

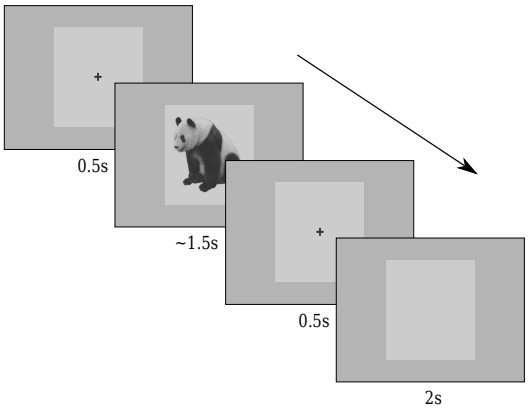


Figure 1: Presentation of image stimuli

Mammals anteater, armadillo, badger, beaver, bison, boar, camel, chamois, chimpanzee, deer, elephant, fox, giraffe, gorilla, hare, hedgehog, hippopotamus, ibex, kangaroo, koala, llama, mole, monkey, mouse, otter, panda, rhinoceros, skunk, squirrel, zebra

Tools Allen key, axe, chainsaw, craft knife, crowbar, file, garden fork, garden trowel, hacksaw, hammer, mallet, nail, paint brush, paint roller, pen knife, pick axe, plaster trowel, pliers, plunger, pneumatic drill, power drill, rake, saw, scissors, scraper, screw, screwdriver, sickle, spanner, tape measure

The EEG signals were recorded at 500Hz from 64 scalp locations based on the 10-20 standard

[1] Using the E-Prime software package: `http://www.pstnet.com/e-prime/`.

montage.[2] The EEG recording computer and stimulus presentation computer were synchronised by means of parallel port transmitted triggers. After the experiment, pre-processing of the recorded signals was carried out using the EEGLAB package (Delorme and Makeig, 2003): signals were band-pass filtered at 1-50Hz to remove slow drifts and high-frequency noise, and then down-sampled to 120Hz. An ICA decomposition was subsequently applied (Makeig et al., 1996), and signal components due to eye-movements were manually identified and removed.

As a preliminary test to verify that the recorded signals included category specific patterns, we applied a discriminative classification technique based on source-separation, similar to that described in Murphy et al. (2008). This found that the categories of mammals and tools could be distinguished with an accuracy ranging from 57% to 80% (mean of 72% over the seven participants).

2.2 Feature extraction

The features extracted are metrics of signal power at a particular scalp location, in a particular frequency band, and at a particular time latency relative to the presentation of each image stimulus. Termed Event Related Synchronisation (ERS) or Event Related Spectral Perturbation (ERSP), such frequency-specific changes in signal amplitude are known to correlate with a wide range of cognitive functions (Pfurtscheller and Lopes da Silva, 1999), and have specifically been shown to be sensitive to category distinctions during the processing of linguistic and visual stimuli (Murphy et al., 2008; Gilbert et al., 2009).

Feature extraction and selection is performed individually on a per-participant basis. As a first step all signal channels are z-score normalised to control for varying conductivity at each electrode site, and a Laplacian sharpening is applied to counteract the spatial blurring of signals caused by the skull, and so minimise redundancy of information between channels.

For each stimulus presentation, 14,400 signal power features are extracted: 64 electrode channels by 15 frequency bands (of width 3.3Hz, between 1 and 50Hz) by 15 time intervals (of length 67ms, in the first second after image presentation). A z-score normalisation is carried out across all stimulus presentations to equalise variance across frequencies and times: to control both for the low-pass filtering action of the skull, and for the reduced synchronicity of activity at increasing latencies. For each stimulus a mean is then taken over each of six presentations to arrive at a more reliable power estimate for each feature.[3]

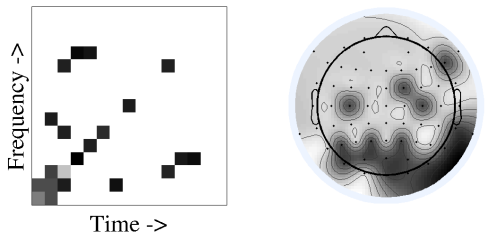


Figure 2: Mean rank of selected features in the time/frequency space (left panel) and on the scalp (right panel) for participant E

The feature ranking method used in Mitchell et al. (2008) evaluates the extent to which the relationship among stimuli is stable across across presentations, using a correlational measure,[4] but preliminary analyses with this selection method on EEG features proved disappointing. Here, two additional ranking criteria are used: each feature is evaluated for its noisiness (the amount of power variation seen across presentations of the same stimulus), and for its distinctiveness (the amount of variation in power estimates across different stimuli). A combination of these three strategies is used to rank the features by their informativeness, and the top 50 features are then selected for each participant.[5]

A qualitative evaluation of the feature selection strategy can be carried out by examining the distribution of features selected. Figure 2 shows the distribution of selected features over the time/frequency spectrum (left panel), and over the scalp (right panel - viewed from above, with the nose pointing upwards). The distribution seen is

[2]Using a Brain Vision BrainAmp system: `http://www.brainproducts.com/`.

[3]Stimulus power features are isolated by band-pass filtering for the required frequencies, cropping following the relevant time interval relative to each image presentation, and then taking the variance of the resulting signal, which is proportional to power.

[4]See the associated supplementary materials of Mitchell et al. (2008) for details: `http://www.sciencemag.org/cgi/content/full/320/5880/1191/DC1`.

[5]Several combinations of these parameters (selection thresholds of 5, 20, 50, 100, 200 features; ranking criteria in isolation and in combination) were investigated - the one chosen gave highest overall performance with the web-derived corpus model: 50 features, combined ranking criteria.

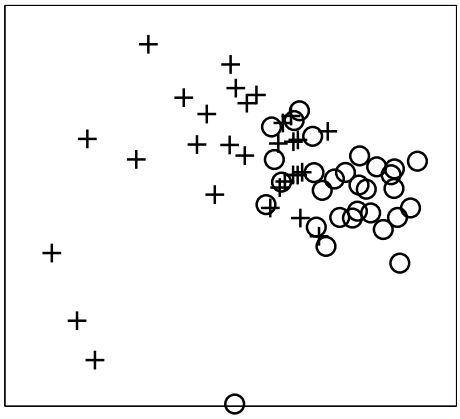


Figure 3: First two components of principal components analysis of selected features for participant E (crosses: mammals; circles: tools)

plausible in reference to previous work: lower frequencies (Pfurtscheller and Lopes da Silva, 1999), latencies principally in the first few hundred milliseconds (Kiefer, 2001), and activity in the visual centres at the rear of the head, as well as parietal areas (Pulvermüller, 2005). A principal components analysis can also be performed on the selected features to see if they reflect any plausible semantic space. As figure 3 shows, the feature selection stage has captured quite faithfully the mammal/tool distinction in a totally unsupervised fashion.

3 Corpus-based semantic models

Data from linguistics (Pustejovsky, 1995; Fillmore, 1982) and neuroscience (Barsalou, 1999; Barsalou, 2003; Pulvermüller, 2005) underline how certain verbs, by emphasising typical ways in which we interact with entities and how they behave, are pivotal in the representation of concrete nominal concepts. Following these traditions, Mitchell et al. (2008) use 25 manually picked verbs as their corpus-based features.

Here that approach is replicated by translating these verbs into Italian. Mitchell et al. (2008) selected verbs that denote our interaction with objects and living things, such as *smell* and *ride*. While the translations are not completely faithful (because frequent verbs of this sort tend to span different sets of senses in the two languages), the aim was to respect the same principle when building the Italian list. The full list, with our back translations into English, is presented in Table 1. We refer to this set as the "Mitchell" verbs.

alzare "raise"	annusare "smell/sniff"
aprire "open"	ascoltare "listen"
assaggiare "taste"	avvicinare "near"
cavalcare "ride"	correre "run/flow"
dire "say/tell"	entrare "enter"
guidare "drive"	indossare "wear"
maneggiare "handle"	mangiare "eat"
muovere "move"	pulire "clean"
puzzare "stink"	riempire "fill"
rompere "break"	sentire "feel/hear"
sfregare "rub"	spingere "push"
temere "fear"	toccare "touch"
vedere "see"	

Table 1: The "Mitchell" verbs, with English translations

As in Mitchell et al. (2008), in order to find a corpus large enough to provide reliable co-occurrence statistics for our target concepts and the 25 verbs, we resorted to the Web, queried using the Yahoo! API.[6] In particular, we represent each concept by a vector that records how many times it co-occurred with each target verb within a span of 5 words left and right, according to Yahoo! counts. We refer to this corpus-based model as the *yahoo-mitchell* model below.

While manual verb picking has proved effective for Mitchell and colleagues (and for us, as we will see in a moment), ultimately what we are interested in is discovering the most distinctive features of each conceptual category. We are therefore interested in more systematic approaches to inducing corpus-based concept descriptions, and in which of these approaches works best for this task. The alternative models we consider were not extracted from the Web, but from an existing corpus, so that we could rely on pre-existing linguistic annotation (POS tagging, lemmatization, dependency paths), and perform more flexible, annotation-aware queries to collect co-occurrence statistics.

More specifically, we used the *la Repubblica/SSLMIT* corpus[7], that contains about 400 million tokens of newspaper text. From this, we extracted four models where nominal concepts are represented in terms of patterns of co-occurrence with verbs (we collected statistics for the top 20,000 most common nouns in the corpus, including the concepts used as stimuli in the silent nam-

[6] http://developer.yahoo.com/search/
[7] http:://sslmit.unibo.it/repubblica/

ing experiment, and the top 5,000 verbs). We first re-implemented a "classic" window-based word space model (Sahlgren, 2006), referred to below as *repubblica-window*, where each noun lemma is represented by its co-occurrence with verb lemmas within the maximum span of a sentence, with no more than one other intervening noun. The *repubblica-position* model is similar, but it also records the position of the verb with respect to the noun (so that *X-usare* "X-use" and *usare-X* "use-X" count as different features), analogously to the seminal HAL model (Lund and Burgess, 1996). It has been shown that models that take the syntactic relation between a target word and a collocate feature into account can outperform "flat" models in some tasks (Padó and Lapata, 2007). The next two models are based on the dependency parse of the *la Repubblica* corpus documented by Lenci (2009). We only counted as collocates those verbs that were linked to nouns by a direct path (such as subject and object) or via preposition-mediated paths (e.g., *tagliare* ***con*** *forbici* "to cut **with** scissors"), and where the paths were among the top 30 most frequent in the corpus. In the *repubblica-depfilter* model, we record co-occurrence with verbs that are linked to the nouns by one of the top 30 paths, but we do not preserve the paths themselves in the features. This is analogous to the model proposed by Padó and Lapata (2007). In the *repubblica-deppath* model, we preserve the paths as part of the features (so that *subj-uccidere* "subj-kill" and *obj-uccidere* count as different features), analogously to Lin (1998), Curran and Moens (2002) and others. For all models, following standard practice in computational linguistics (Evert, 2005), we transform raw co-occurrence counts into log-likelihood ratios.

Following the evaluation paradigm of Mitchell et al. (2008), linear models trained on corpus-based features are used to predict the pattern of EEG activity for unseen concepts. This only works if we have a very limited number of features (or else we would have more parameters to estimate than data-points to estimate them). The Repubblica-based models have thousands of features (one per verb collocate, or verb+path collocate). We adopt two strategies to select a reduced number of features. In the *topfeat* versions, we first pick the 50 features that have the highest association with each of the target concepts. We then count in how many of these concept-specific top lists a feature occurs, and we pick the 25 features that occur in the largest number of them. The intuition is that this should give us a good trade-off between how characteristic the features are (we only use features that are highly associated with some of our concepts), and their generalization capabilities (we pick features that are associated with multiple concepts). Randomly selected examples of the features extracted in this way for the various Repubblica models are reported in Table 2.

repubblica-window	*repubblica-position*
abbattere "demolish"	X-ferire "X-wound"
afferrare "seize"	X-usare "X-use"
impugnare "grasp"	dipingere-X "paint-X"
tagliare "cut"	munire-X "supply-X"
trovare "find"	tagliare-X "cut-X"
repubblica-depfilter	*repubblica-deppath*
abbattere "demolish"	con+tagliare "with+cut"
correre "run"	obj+abbattere "obj+demolish"
parlare "speak"	obj+uccidere "obj+kill"
saltare "jump"	intr-subj+vivere "intr-subj+live"
tagliare "cut"	tr-subj+aprire "tr-subj+open"

Table 2: Examples of top features from the *la Repubblica* models

Alternatively, instead of feature *selection* we perform feature *reduction* by means of a Singular Value Decomposition (SVD) of the noun-by-verb matrix. We apply the SVD to matrices that include the top 20,000 most frequent nouns in the corpus (including our target concepts) since the quality of the resulting reduced model should improve if we can exploit richer patterns of correlations among the columns – verbs – across rows – nouns (Landauer and Dumais, 1997; Schütze, 1997). In the *svd* versions of our models, we pick as features the top 25 left singular vectors, weighted by the corresponding singular values. These features do not have a straightforward interpretation, but they tend to group verb meanings that belong to broad semantic domains. For example, among the original verbs that are most strongly correlated with one of the top singular vectors of *repubblica-window* we find *giocare* "play", *vincere* "win" and *perdere* "lose". Another singular vector is associated with *ammontare* "amount", *costare* "cost", *pagare* "pay", etc. One of the top singular vectors of *repubblica-deppath* is strongly correlated with *in+scendere* "descend into", *in+mettere* "put into", *in+entrare* "enter into", though not all singular vectors are so clearly characterized by the verbs they correlate with.

None of the *la Repubblica* models had full coverage of our concept stimulus set (see the second column of Table 3 below), because our extraction method missed some multi-word units, and feature selection led to losing some more items due to data sparseness (e.g., some target words had no collocates connected by the dependency paths we selected). The experiments reported in the next section used all the target concepts available in each model, but a replication using the 50 concepts that were common to all models obtained results that are comparable. For a direct comparison between Yahoo! and *la Repubblica* derived features, we tried collecting statistics for the Mitchell verbs from Repubblica as well, but the resulting model was extremely sparse, and we do not report its performance here.

Finally, it is important to note that any representation yielded by a corpus semantic model does not characterise a concept directly, but is rather an aggregate of the various senses and usages of the noun chosen to represent it. This obvious limitation will persist until comprehensive, robust and computationally efficient word-sense disambiguation techniques become available. However these models are designed to extract semantic (as opposed to syntactic or phonological) properties of words, and as noted in the introduction, have been demonstrated to correlate with behavioural effects of conceptual processing.

4 Predicting EEG patterns using corpus-based models

In Section 2.2 above we showed how we extracted features summarizing the spatial, temporal and frequency distribution of the EEG signal collected while participants were processing each of the target concepts. In Section 3, we described various ways to obtain a compact representation of the same concepts in terms of corpus-derived features. We will now discuss the method we employed to verify whether the corpus-derived features can be used to predict the EEG patterns – that is whether semantics can be used to predict neural activity. Our hope is that a good corpus-based model will provide a decomposition of concepts into meaningful properties, corresponding to coherent subpatterns of activation in the brain, and thus capture generalizations across concepts. For example, if a concept is particularly visually evocative (e.g., *zebra*), we might expect it to be strongly associated with the verb *see*, while also causing particular activation of the vision centres of the brain. Similarly, concepts with strong associations with a particular sound (e.g., *cuckoo*) might be semantically associated with *hear* while also disproportionately activating auditory areas of the brain. It should thus be possible to learn a model of corpus-to-EEG-pattern correspondences on training data, and use it to predict the EEG activation patterns of unseen concepts.

We follow the paradigm proposed by Mitchell et al. (2008) for fMRI data. For each participant and selected EEG feature, we train a model where the level of activation of the latter in response to different concepts is approximated by a linear combination of the corpus features:

$$\vec{f} = \mathbf{C}\vec{\beta} + \vec{\epsilon}$$

where $\vec{f}$ is the vector of activations of a specific EEG feature for different concepts, the matrix $\mathbf{C}$ contains the values of the corpus features for the same concepts (row-normalised to z-scores), $\vec{\beta}$ is the weight we must learn for each corpus feature, and $\vec{\epsilon}$ is a vector of error terms. We use the method of least squared errors to learn the weights that maximize the fit of the model. We can then predict the activation of an EEG feature in response to a new concept that was not in the training data by a $\vec{\beta}$-weighted sum of the values of each corpus feature for the new concept. In some cases collinearity in the corpus data (regular linear relationships among the corpus-feature columns) prevented the estimation procedure from finding a solution. In such cases (due to the small number of data, relative to the number of unknowns), the least informative corpus-features (those that correlated on average most highly with other features) were iteratively removed until a solution was reached. All models were trained with between 23 and 25 corpus features.

Again following Mitchell and colleagues, we adopt a leave-2-out paradigm in which a linear model for each EEG feature is trained in turn on all concepts minus 2. For each of the 2 left out concepts, we predict the EEG activation pattern using the trained linear model and their corpus features, as just described. We then try to correctly match the predicted and observed activations, by measuring the Euclidean distance between the model-generated EEG activity (a vector of estimated power levels for the n EEG fea-

tures selected) and the corresponding EEG activity recorded in the experiment (other distance metrics gave similar results to the ones reported here). Given the 2 left-out concepts *a* and *b*, we compute 2 *matched* distances (i.e., distance between predicted and observed pattern for *a*, and the same for *b*) and 2 *mismatched* distances (predicted *a* and observed *b*; predicted *b* and observed *a*). If the average of the matched distances is lower than the average of the mismatched distances, we consider the prediction successful – otherwise we count is as a failed prediction. At chance levels, expected matching accuracy is 50%.

5 Results

Table 3 shows the comparative results for all the corpus models introduced in Section 3. The third column (*all*) shows the overall accuracy in correctly matching predicted to observed EEG activity patterns, and so successfully distinguishing word meanings. The significance of the figures is indicated with the conventional annotation (calculated using a one-way two-sided t-test across the individual participant accuracy figures against an expected population mean of 50%).[8] The second column shows the coverage of each model of the 60 mammal and tool concepts used, which ranged from full (for the *yahoo-mitchell* model) to 51 concepts (for the *depfilter-topfeat* model). The corresponding number of matching comparisons over which accuracy was calculated ranged from 1770 down to 1225.

As suggested by previous work (Murphy et al., 2008), and illustrated by figure 3, coarse distinctions between words in different superordinate categories (e.g., *hammer* vs *armadillo*; *giraffe* vs *nail*) may be easier to detect than those among concepts within the same category (e.g., *hammer* vs *nail*; *giraffe* vs *armadillo*). The fourth and fifth columns give these accuracies, and while between-category discriminations do prove more reliable, they indicate that, for the top rated model at least, finer within-category distinctions are also being captured. Figures from the top two performing models are given for individual participants in tables 4 and 5.

[8]On average, the difference seen between matched and mismatched pairs was small, at about 3% of the distance between observed and predicted representations, and was marginally bigger for correct than for incorrect predictions ($p < 0.01$).

part.	*overall*	*within*	*between*
A	54	53	55
B	54	47	60
C	62	56	67
D	61	56	67
E	68	58	78
F	52	54	51
G	57	51	63

Table 4: Accuracy levels for individual participant sessions, *yahoo-mitchell* web corpus

part.	*overall*	*within*	*between*
A	49	52	46
B	59	57	60
C	60	60	59
D	50	45	55
E	56	53	58
F	64	64	65
G	52	49	55

Table 5: Accuracy levels for individual participant sessions, *repubblica-window-svd*

6 Discussion

Our results show that corpus-extracted conceptual models can be used to distinguish between the EEG activation levels associated with conceptual categories to a degree that is significantly above chance. Though category specific patterns are detectable in the EEG signal alone (as illustrated by the PCA analysis in figure 3), on that basis we cannot be sure that semantics is being detected. Some other property of the stimuli that co-varies with the semantic classes of interest could be responsible, such as visual complexity, conceptual familiarity, lexical frequency, or phonological form. Only by cross-training with individual corpus features and showing that these hold a predictive relationship to neural activity have we been able to establish that EEG patterns encode semantics.

Present evidence indicates that fMRI may provide richer data for training such models than EEG (Mitchell and colleagues obtain an average accuracy of 77%, and 65% for the within category setting). However, fMRI has several clear disadvantages as a tool for language researchers. First of all, the fine spatial resolution it provides (down to 2-3mm), while of great interest to neuroscientists, is not in itself linguistically informative. Its coarse temporal resolution (of the order of several seconds), makes it ill-suited to analysing on-line linguistic processes. EEG on the other hand, despite its low spatial resolution (several centimetres), gives millisecond-level temporal resolution,

model	*coverage*	*all*	*within cat*	*between cat*
yahoo-mitchell	100	58.3** (5.7)	53.6* (3.7)	63.0** (8.9)
repubblica-window-svd	96.7	55.7* (5.6)	54.3 (6.5)	56.9* (5.9)
repubblica-window-topfeat	93.3	52.1 (4.3)	48.7 (3.6)	55.4 (7.0)
repubblica-deppath-svd	93.3	51.4 (8.7)	49.0 (8.0)	54.0 (10.0)
repubblica-depfilter-topfeat	85.0	51.1 (9.6)	49.3 (9.6)	53.1 (10.0)
repubblica-position-topfeat	93.3	50.0 (5.2)	46.0 (4.7)	53.6 (8.0)
repubblica-deppath-topfeat	86.7	49.9 (9.0)	47.0 (9.3)	52.4 (9.6)
repubblica-position-svd	96.7	49.4 (10.2)	46.6 (9.8)	52.3 (11.3)
repubblica-depfilter-svd	93.3	48.9 (11.1)	47.1 (8.9)	50.6 (12.9)

Table 3: Comparison across corpus models, with percentage concept coverage, mean cross-subject percentage prediction accuracy and standard deviation; $*p < 0.05$, $**p < 0.01$

enabling the separate analysis of sequential cognitive processes and states (e.g., auditory processing, word comprehension, semantic representation). fMRI is also prohibitively expensive for most researchers (ca. 300 euros per hour at cost price), compared to EEG (ca. 30 euros per hour). Finally, there is no prospect of fMRI being miniaturised, while wearable EEG systems are already becoming commercially available, making experimentation in more ecological settings a possibility (e.g., playing with a child, meeting at a desk, walking around). In short, while EEG can be used to carry out systematic investigations of categorical distinctions, doing so with fMRI would be prohibitively expensive.

Present results indicate that distinctions between categories are easier than distinctions between category elements; and that selecting the conceptual features by hand gives better results than discovering them automatically. Both of these results however may be due to limitations of the current method. One limitation is that we have been using the same set of features for all concepts, which is likely to blur the distinctions between members of a category more than those between categories. A second limitation of our present methodology is that it is constrained to use very small numbers of semantic features, which limits its applicability. For example it is hard to conceive of a small set of verbs, or other parts-of-speech, whose co-occurrence patterns could successfully characterise the full range of meaning found in the human lexicon. Even the more economical corpus-extracted conceptual models tend to run in the hundreds of features (Almuhareb, 2006). We are currently working on variations in the method that will address these shortcomings.

The web-based model with manually picked features outperformed all *la Repubblica*-based models. However, the results attained with *repubblica-window-svd* are encouraging, especially considering that we are reporting results for an EEG feature configuration optimised for the web data (see footnote 5), and that *la Repubblica* is several orders of magnitude smaller than the web. That data sparseness might be the main issue with *la Repubblica* models is suggested by the fact that *repubblica-window-svd* is the least sparse of them, since it does not filter data by position or dependency path, and compresses information from many verbs via SVD. In future research, we plan to extract richer models from larger corpora. And as the discriminative accuracy of cross-training techniques improves, further insights into the relative validity of corpus representations will be attainable. One research aim is to see if individual corpus semantic properties are encoded neurally, so providing strong evidence for a particular model. These techniques may also prove more objective and reliable in evaluating representations of abstract concepts, for which it is more difficult to collect reliable judgements from informants.

References

A. Almuhareb. 2006. *Attributes in lexical acquisition*. Dissertation, University of Essex.

L. Barsalou. 1999. Perceptual symbol systems. *Behavioural and Brain Sciences*, 22:577–660.

L. Barsalou. 2003. Situated simulation in the human conceptual system. *Language and Cognitive Processes*, 18:513–562.

J.R. Curran and M. Moens. 2002. Improvements in automatic thesaurus extraction. In *Proceedings of SIGLEX*, pages 59–66.

A. Delorme and S. Makeig. 2003. Eeglab: an open source toolbox for analysis of single-trial dynamics includingindependent component analysis. *Journal of Neuroscience Methods*, 134:9–21.

S. Evert. 2005. *The statistics of word cooccurrences*. Dissertation, Stuttgart University.

Ch. J. Fillmore. 1982. Frame semantics. In Linguistic Society of Korea, editor, *Linguistics in the Morning Calm*, pages 111–138. Hanshin, Seoul.

J. Gilbert, L. Shapiro, and G. Barnes. 2009. Processing of living and nonliving objects diverges in the visual processing system: evidence from meg. In *Proceedings of the Cognitive Neuroscience Society Annual Meeting*.

M. Kiefer. 2001. Perceptual and semantic sources of category-specific effects in object categorization:event-related potentials during picture and word categorization. *Memory and Cognition*, 29(1):100–116.

T. Landauer and S. Dumais. 1997. A solution to Platos problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

A. Lenci. 2009. Argument alternations in italian verbs: a computational study. In *Atti del XLII Congresso Internazionale di Studi della Società di Linguistica Italiana*.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *COLING-ACL98*, Montreal, Canada.

K. Lund and C. Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, 28:203—208.

S. Makeig, A.J. Bell, T. Jung, and T.J. Sejnowski. 1996. Independent component analysis of electroencephalographic data. In *in Advances in Neural Information Processing Systems*, pages 145–151. MIT Press.

A. Martin and L. Chao. 2001. Semantic memory and the brain: structure and processes. *Current Opinions in Neurobiology*, 11:194–201.

T. Mitchell, S. Shinkareva, A. Carlson, K. Chang, V. Malave, R. Mason, and M. Just. 2008. Predicting human brain activity associated with the meanings of nouns. *Science*, 320:1191–1195.

B. Murphy, M. Dalponte, M. Poesio, and L. Bruzzone. 2008. Distinguishing concept categories from single-trial electrophysiological activity. In *Proceedings of the Annual Meeting of the Cognitive Science Society*.

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

G. Pfurtscheller and F. Lopes da Silva. 1999. Event-related EEG/MEG synchronization and desynchronization: Basic principles. *Clinical Neurophysiology*, 110:1842–1857.

F. Pulvermüller. 2002. *The neuroscience of language: on brain circuits of words and serial order*. Cambridge University Press, Cambridge.

F. Pulvermüller. 2005. Brain mechanisms linking language and action. *Nature Reviews Neuroscience*, 6:576–582.

J. Pustejovsky. 1995. *The Generative Lexicon*. MIT Press, Cambridge.

M. Sahlgren. 2006. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Dissertation, Stockholm University.

S. Schulte im Walde. 2008. *Theoretical adequacy, human data and classification approaches in modelling word properties, word relatedness and word classes*. Habilitation, Saarland University.

H. Schütze. 1997. *Ambiguity Resolution in Natural Language Learning*. CSLI, Stanford.

A Comparison of Windowless and Window-Based Computational Association Measures as Predictors of Syntagmatic Human Associations

Justin Washtell
School of Computing
University of Leeds
washtell@comp.leeds.ac.uk

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

Abstract

Distance-based (windowless) word association measures have only very recently appeared in the NLP literature and their performance compared to existing windowed or frequency-based measures is largely unknown. We conduct a large-scale empirical comparison of a variety of distance-based and frequency-based measures for the reproduction of syntagmatic human assocation norms. Overall, our results show an improvement in the predictive power of windowless over windowed measures. This provides support to some of the previously published theoretical advantages and makes windowless approaches a promising avenue to explore further. This study also serves as a first comparison of windowed methods across numerous human association datasets. During this comparison we also introduce some novel variations of window-based measures which perform as well as or better in the human association norm task than established measures.

1 Introduction

Automatic discovery of semantically associated words has attracted a large amount of attention in the last decades and a host of computational association measures have been proposed to deal with this task (see Section 2). These measures traditionally rely on the co-ocurrence frequency of two words in a corpus to estimate a relatedness score. There has been a recent emergence of distance-based language modelling techiques in NLP (Savicki and Hlavacova, 2002; Terra and Clarke, 2004) in which the number of tokens separating words is the essential quantity. While some of this work has considered distance-based alternatives to conventional association measures (Hardcastle, 2005; Washtell, 2009), there has been no principled empirical evaluation of these measures as predictors of human association. We remedy this by conducting a thorough comparison of a wide variety of frequency-based and distance-based measures as predictors of human association scores as elicited in several different *free word association tasks*.

In this work we focus on first-order association measures as predictors of *syntagmatic* associations. This is in contrast to second and higher-order measures which are better predictors of paradigmatic associations, or word *similarity*. The distinction between syntagmatic and paradigmatic relationship types is neither exact nor mutually exclusive, and many paradigmatic relationships can be observed syntagmatically in the text. Roughly in keeping with (Rapp, 2002), we hereby regard *paradigmatic* assocations as those based largely on word similarity (i.e. including those typically classed as synonyms, antonyms, hypernyms, hyponyms etc), whereas *syntagmatic* associations are all those words which strongly invoke one another yet which cannot readily be said to be similar. Typically these will have an identifiable semantic or grammatical relationship (meronym/holonym: *stem – flower*, verb/object: *eat – food* etc), or may have harder-to-classify topical or idiomatic relationships (*family – Christmas*, *rock – roll*).

We will show in Section 3.2 that syntagmatic relations by themselves constitute a substantial 25-40% of the strongest human responses to cue words. Although the automatic detection of these assocations in text has received less attention than that of paradigmatic associations, they are nonetheless important in applications such as the resolution of bridging anaphora (Vieira and Poesio, 2000).[1] Furthermore, first-order associations

[1] where for example resolving *my house – the windows* to *the windows of my house* can be aided by the knowledge that windows are often (syntagmatically) associated with houses.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 628–637,
Singapore, 6-7 August 2009.

are often the basis of higher-order vector word-space models used for predicting paradigmatic relationships: i.e. through the observation of words which share similar sets of syntagmatic associations. Therefore improvements made at the level we are concerned with may reasonably be expected to carry through to applications which hinge on the identification of paradigmatic relationships.

After a discussion of previous work in Section 2, we formulate the exact association measures and parameter settings which we compare in Section 3, where we also introduce the corpora and human association sets used. Then, by using evaluations similar to those described in (Baroni et al., 2008) and by Rapp (2002), we show that the best distance-based measures correlate better overall with human association scores than do the best window based configurations (see Section 4), and that they also serve as better predictors of the strongest human associations (see Section 5).

2 Related Work

Measures based on co-ocurrence frequency. The standard way of estimating the syntagmatic association of word pairs in a corpus is to examine the frequency of their co-occurence, and then usually to compare this to some expected frequency. There are a host of measures which exist for this purpose. After raw co-occurrence frequency, the simplest and most prevalent in the literature is Pointwise Mutual Information, famously used by Church (1989) (as the *association ratio*). This is defined as the log of the ratio of the observed co-occurrence frequency to the frequency expected under independence. More sophisticated and statistically-informed measures include t-Score, z-Score, Chi-Squared and Log-Likelihood (see Evert (2005) for a thorough review).

All of these measures have in common that they require co-occurrence frequency to be specified, and therefore require some definition of a region within which to count co-occurrences. This region might be the entirety of a document at one extreme, or a bigram at the other. A versatile and hugely popular generalised approach is therefore to consider a "'window"' of w words, where w can be varied to suit the application. Unsurprisingly, it has been found that this is a parameter which can have a significant impact upon performance (Yarowsky and Florian, 2002; Lamjiri et al., 2004; Wang, 2005). While choosing an optimum window size for an application is often subject to trial and error, there are some generally recognized trade-offs between small versus large windows, such as the impact of data-sparseness, and the nature of the associations retrieved (Church and Hanks, 1989; Church and Hanks, 1991; Rapp, 2002)

Measures based on distance between words in the text. The idea of using distance as an alternative to frequency for modelling language has been touched upon in recent literature (Savicki and Hlavacova, 2002; Terra and Clarke, 2004; Hardcastle, 2005). Washtell (2009) showed that it is possible to build distance-based analogues of existing syntagmatic association measures, by using the notions of mean and expected distance rather than of frequency. These measures have certain theoretical qualities - notably scale-independence and relative resilience to data-sparseness - which might be expected to provide gains in tasks such as the reproduction of human association norms from corpus data. The specific measure introduced by Washtell, called Co-Dispersion, is based upon an established biogeographic dispersion measure (Clark and Evans, 1954). We provide a thorough empirical investigation of Co-Dispersion and some of its derivatives herein.

Measures based on syntactic relations. Several researchers (Lin, 1998; Curran, 2003; Pado and Lapata, 2007) have used word space models based on grammatical relationships for detecting and quantifying (mostly paradigmatic) word associations. In this paper, we will not use syntactic relation measures for two main reasons. Firstly these depend on the availability of parsers, which is not a given for many languages. Secondly, this may not be the most pertinent approach for predicting human free associations, in which certain observed relationsips can be hard to express in terms of syntactic relationships.

3 Methodology

Similar to (Rapp, 2002; Baroni et al., 2008, among others), we use comparison to human assocation datasets as a test bed for the scores produced by computational association measures. An alternative might be to validate scores against those derived from a structured resource such as WordNet.

Table 1: Human association datasets

Name	Origin	Cues	Respondents
Kent	Kent & Rosanoff (1910)	100	$\sim$ 1000
Minnesota	Russell & Jenkins (1954)	100	$\sim$ 1000
EAT	Kiss et al (1973)	8400	100
Florida	Nelson et al (1980)	5019	$\sim$ 140

However, relatedness measures for WordNet are many and varied and are themselves the subject of evaluation (Pedersen et al., 2004). Although human association datasets have their own peculiarities, they do at least provide some kind of definite Gold Standard. Yet another alternative might be to incorporate our computational association scores into an application (such as anaphora resolution), and measure the performance of that, but noise from other submodules would complicate evaluation. We leave such extensions to possible future work.

We use evaluations similar to those used before (Rapp, 2002; Pado and Lapata, 2007; Baroni et al., 2008, among others). However, whereas most existing studies use only one dataset, or hand-selected parts thereof, we aim to evaluate measures across four different human datasets. In this way we hope to get as unbiased a picture as possible.

3.1 Association data

The datasets used are listed in Table 1. While the exact experimental conditions may differ, the datasets used were all elicited using the same basic methodology: by presenting individual words (*cues*) to a number of healthy human subjects and asking in each case for the word that is most immediately or strongly evoked. An association score can then be derived for each cue/response pair in a dataset by dividing the number of participants providing a given response by the number who were presented with the cue word. In Table 1, *respondents* refers to the number of people from whom a response was solicited for each cue word in a study (this is not to be confused with the number of unique responses).

Of these four datasets, one (Kent & Rosanoff) appears not to have been previously used in any peer-reviewed study of corpus-derived lexical association. It is worth noting that some of these datasets are quite dated, which might affect correlations with corpus-derived scores, as culture and contemporary language have a fundamental impact upon the associations humans form (White and Abrams, 2004).

3.2 Frequency of Syntagmatic Associations

To verify that strong human associations do include a large number of syntagmatic associations, we manually annotated all pairs consisting of a cue and its strongest human response in the `Minnesota` and `Kent` datasets as expressing either a syntagmatic or a paradigmatic relationship. The overall set to be annotated consisted of 200 pairs.

Annotators were given short (half-page) guidelines on syntagmatic and paradigmatic assocations, stating that very similar items (including hyponyms/hypernyms) as well as antonyms were to be judged as paradigmatic whereas words that do not fulfil this criterion are to be judged as syntagmatic. The two annotators were the authors of this paper (one native and one near-native speaker). After independent annotation, agreement was measured at a percentage agreement of 91/93% and a kappa of 0.80/0.82 for `Minnesota` and `Kent`, respectively. Therefore, the distinction can be made with high reliability.

Overall, 27/39% of the human responses were syntagmatic in the Kent/Minnesota datasets, showing that syntagmatic relations make up a large proportion of even the strongest human associations.

3.3 Corpora

We use two randomized subsets of the British National Corpus (BNC), a representative 100 million word corpus of British English (Burnard, 1995): one 10 million word sample, and a 1 million word sample. A vocabulary of approximately 33,000 word types was used. The selected words included approximately 24,000 word types comprising all cue and target words from the multiple sets of human association norms to be used in this study. To these were added a top-cut of the most frequent words in the BNC, until the total of 33,000 word types was reached. The resultant set included ap-

proximately the 24,000 most common word types in the BNC, with the remaining 9000 words types therefore comprising relatively uncommon words taken from the human associative responses.

The words included in the vocabulary accounted for over 94.5% of tokens in the corpus. Although statistics for the remaining word types in the BNC were not gathered, their corresponding tokens were left in the corpus so that these could be properly accounted for when calculating distances and window spans.

In order to maximize matching between word types in the corpus and association norms, all words in both were normalized by converting to lower-case and removing hyphens and periods. Words consisting entirely of numerals, or numerals and punctuation, and all "phrasal" associative responses (those containing spaces) were discarded. The 33,000 word count was satisfied after making these normalizations.

In order to maximize the variety of the language in the samples, the subsets were built from approximately the first 2000 words only of each randomly selected document from the BNC (a similar strategy to that used in constructing the 1 million word Brown Corpus). Both a 10 million word and a 1 million word sample were constructed in this fashion, allowing us to also examine the effects of varying corpus size and content.

3.4 Association measures used

3.4.1 Frequency-based measures

In the following, x is the cue word and y a (possible) response word. Therefore $p(x)$ is the probability of observing x, and $p(\bar{x})$ refers to the probability of not observing x.

Pointwise Mutual Information (hereonin PMI) was introduced in Section 2. For ranking word pairs, we can neglect the usual logarithm.

$$PMI = \frac{p(x,y)}{p(x)p(y)}$$

PMI is infamous for its tendency to attribute very high association scores to pairs involving low frequency words, as the denominator is small in such cases, even though the *evidence* for association in such cases is also small. This can result in some unlikely associations. There exist a number of alternative measures which factor in the amount of evidence to give an estimate of the *significance of association*. One popular and statistically appealing such measure is Log-Likelihood (LL) (Dunning, 1993). LL works on a similar principle to PMI but considers the ratio of the observed to expected co-occurrence frequencies for *all contingencies* (i.e. including those where the words do not co-occur). LL, as it most frequently appears in the literature, is not actually a measure of positive association: it also responds to significant *negative* association. Therefore LL is arguably not suited to the task in hand. Krenn & Evert (2001) experiment with one-tailed variants of LL and Chi-Squared measures, although they do not define these variants. Here, we construct a one-tailed variant of LL by simply reversing the signs of the terms which respond to negative association.

$$\begin{aligned} LL_{1tail} &= p(x,y)\log\frac{p(x,y)}{p(x)p(y)} - p(x,\bar{y})\log\frac{p(x,\bar{y})}{p(x)p(\bar{y})} \\ &- p(\bar{x},y)\log\frac{p(\bar{x},y)}{p(\bar{x})p(y)} + p(\bar{x},\bar{y})\log\frac{p(\bar{x},\bar{y})}{p(\bar{x})p(\bar{y})} \end{aligned}$$

LL does not have a clear analogue amongst the distance-based measures (introduced in Section 3.4.2), whereas PMI for instance does. We therefore construct variants of PMI and other measures which take the *amount of evidence* into account in a way which can be directly reproduced in the distance domain. For this we borrow from Sackett (2001) who asserts that, all other things being equal, statistical significance is proportional to the square root of the sample size. There are a number of ways one might quantify sample size. We take a consistent approach across the various distance-based and frequency-based measures: we assume sample size to be equivalent to the lesser of the frequencies of the two words as this represents the total number of words available for pairing, with fewer observed pairs therefore being considered to constitute negative evidence.

$$PMI_{sig} = \sqrt{\min(p(x),p(y))}\frac{p(x,y)}{p(x)p(y)}$$

All of the above measures are symmetric. Human associative responses however are not (Michelbacher et al., 2007): a person's tendency to give the response *because* to the cue *why* does not necessarily reflect their tendency to give the response *why* to the cue *because*.[2] A simple asymmetric association measure is conditional probability (CP)

[2]This notion of assymmetry is not to be confused with

- the probability of observing the response, given that the cue has already occurred.

$$CP = p(y|x) = \frac{p(x,y)}{p(x)}$$

CP suffers from the fact that it does not account at all for the general frequency of the response word. It therefore tends to favour very frequent words, such as function words. An obvious solution would be to divide CP by the frequency of the response word, however this merely results in PMI which is symmetric. By multiplying CP with PMI (and taking the root, to simplify) we obtain a measure which is asymmetric yet does not overtly favour frequent response words.[3] We refer to this herein as Semi-Conditional Information (SCI).

$$SCI = \frac{p(x,y)}{p(x)\sqrt{p(y)}}$$

We also explore variants of both CP and SCI with the additional significance correction presented for PMI_{sig}. These can be easily inferred from the formulae above.

3.4.2 Distance-based Measures

Co-Dispersion (herein CD), introduced by Washtell (2009), is defined as the ratio of the mean observed distance to the expected distance, where the expected distance is derived from the frequency of the more frequent word type. Distance refers to the number of tokens separating an occurrence of one word and the *nearest* occurrence of another word. Pairs spanning an intervening occurrence of *either* word type or a document boundary are not considered. Note that here we specify only the generalised mean M, as we wish to keep the specific choice of mean as a parameter to be explored,

$$CD = \frac{1/\max(p(x),p(y))}{M(dist_{xy1} \ldots dist_{xyn})}$$

where $dist_{xyi}$ is i^{th} observed distance between some occurrence of word type x and its nearest preceding or following occurrence of word type y, and n is the total number of such distances observed (being at most equal to the frequency of the rarer word).

In cases where many occurrences of the less frequent word were not able to be paired, raw CD gives midleading results. This is because unpairable words themselves provide useful negative evidence which CD ignores. A more appropriate measure can be formed in which the mean distance is calculated using the frequency of the less frequent word, regardless of whether this many distances were actually observed. This gives us Neutrally-Weighted Co-Dispersion (NWCD). Note that for convenience, we keep the standard definition of the mean and introduce a correction factor instead.

$$NWCD = \frac{n}{\min(p(x),p(y))} \frac{1/\max(p(x),p(y))}{M(dist_{xy1} \ldots dist_{xyn})}$$

An asymmetric association measure can be formed in a similar manner. Instead of calculating the mean using the frequency of the less frequent word as described above, we explicitly use the frequency of the cue word (which in some cases may actually *exceed* the number of distances observed). This gives us Cue-Weighted Co-Dispersion (CWCD).

$$CWCD = \frac{n}{p(x)} \frac{1/\max(p(x),p(y))}{M(dist_{xy1} \ldots dist_{xyn})} \quad (1)$$

In addition to these measures, we also explore significance-corrected forms NWCD_{sig} and CWCD_{sig}, by introducing the same sample size term employed by PMI_{sig}, CP_{sig} and SCI_{sig}. Again, these can readily be inferred from the existing formulae in the above two sections.

3.5 Co-occurrence Parameters

For frequency-based co-occurrence statistics, the principle parameter is the window size. We will use five window sizes separated by a constant scaling factor, chosen so as to span those most commonly encountered in the literature, with some extension towards the upper end. We use w to represent this parameter, with $w = 2$ implying a window size of +/-2. The parameter values explored

that of *direction* in the text. While the two may correlate, one can find ample counter-examples: *jerky* triggers *beef* more strongly than *beef* triggers *jerky*.

[3] Note that Wettler & Rapp (1993) introduced a more general asymmetric measure for predicting human associations, by employing an exponent parameter to $p(y)$. Our formulisation is equivalent to their measure with an exponent of 0.5, whereas they found an exponent of 0.66 to be most effective in their empirical study. Exponents of 0 and 1 result in CP and PMI respectively.

are $w = 2$, $w = 10$, $w = 50$, $w = 250$ and $w = 1250$. We examine such large window sizes so as to give a fairer comparison with the distance approach which is not bounded by a window, and in acknowledgement of the fact that the entire document as context has been used with some success in other application areas (most notably information retrieval).

For distance-based statistics, the principle parameter is the function via which the various observed distances between tokens are reduced to a single mean value. In this investigation we will explore five means. These are the power means with exponents (which herein we refer to as m) ranging from -2 to +2. These give us the quadratic mean or RMS ($m = 2$), the arithmetic mean ($m = 1$), the geometric mean ($m = 0$), the harmonic mean ($m = -1$), and the inverse quadratic mean ($m = -2$).

4 Task I: Correlations on word pairs

One of the ESSLLI Workshop shared tasks (Baroni et al., 2008) required the evaluation of correlation between a small, manually selected subset of human cue-response scores from the `EAT` dataset and automatic scores for the same word pairs. Here, tather than focusing on word pairs which meet certain grammatical and frequency criteria we test on all pairs. For the `EAT` and `Florida` datasets, this amounts to many tens of thousands of cue-response pairs. Although this makes the task of correlation harder, it means we can attribute a great deal of statistical significance to the results and make our observations as general as possible.

4.1 Evaluation Measures, Upper Bounds and Baselines

For evaluating agreement between corpus-derived associations and human associations, we use Spearman's Rank correlation. This is appropriate because we are primarily interested in the relative ranking of word pair associations (in order to predict particularly strong responses, for example). Although some studies have used Pearson's correlation, the various association measures explored here are not linear within each another and it would be inappropriate to evaluate them under the assumption of a linear relationship with the human norms.

Two of the human datasets, `Kent` and `Minnesota`, though collected independently, are based on the same set of 100 cue words established by Kent (1910). Therefore by performing a rank correlation of these two datasets with one another, (each of which was produced by pooling the responses of some 1000 people) we can get a useful upper-bound for correlations: if a computer-based system were to exceed this upper-bound in correlations with either dataset, then we would need to suspect it of over-fitting.

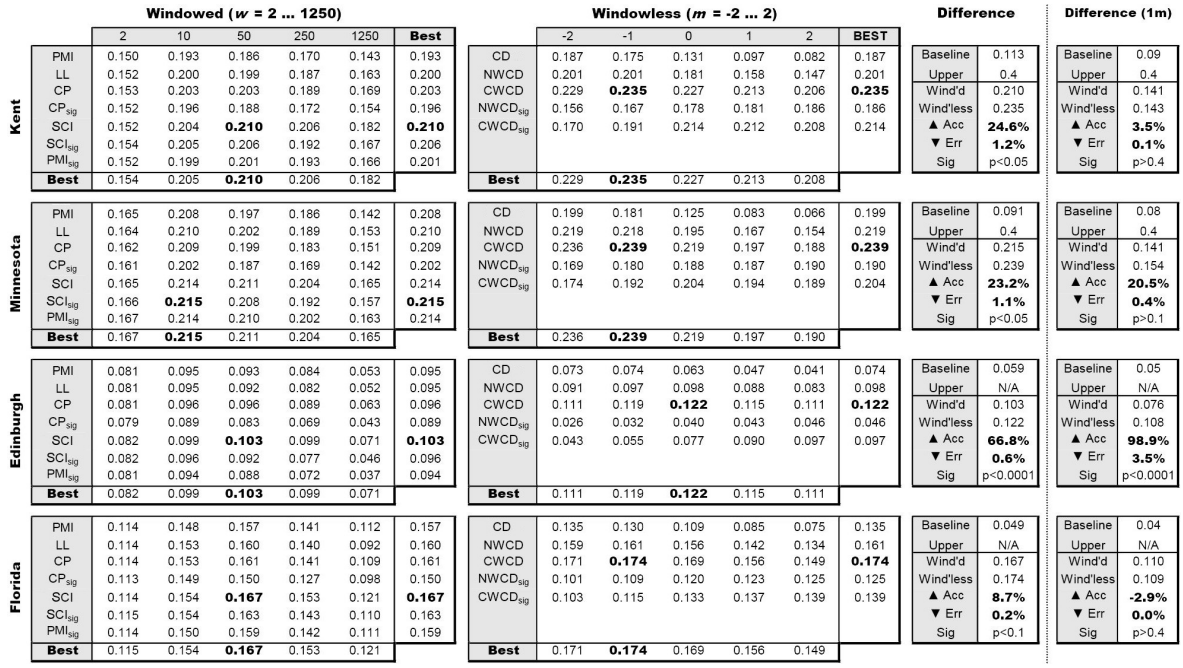
As a baseline, we use the corpus frequency of the response word. The simple assumption is that the more frequent a word is, the more likely it is to appear as a human response independent of the cue given. This is also the simplest formulation which does not assign equal scores to the various possible responses, and which is therefore capable of producing a rank-list of predictions.

4.2 Task I Results

Figure 1 shows the Spearman's rank correlation co-efficients across all paramaterisations of all association measures (frequency-based on the left, and distance-based on the right), with each human dataset, for the 10 million word corpus. Emboldened are the best performing windowed and windowless configurations for each dataset. The difference of these figures over the baseline is highly significant ($p < 0.0001$ in most cases). The panels to the right show summary statistics for these figures, and for the 1 million word corpus (for which full figures are not included owing to space limitations). These statistics include the performance of the baseline, where relevant the estimated upper-bound (see Section 4.1), and the difference in performance of the distance-based method over the window-based. The accuracy and error figures are based on the co-efficients of determination (r^2) and are expressed both as a relative improvement in accuracy (how much closer (r^2) is to 1 under the distance-based approach) and reduction in error (how much further r^2 is from zero). Also the significance of the difference in the r values is given.

4.3 Discussion

The two-way Spearman's rank correlations between the `Kent` and `Minesota` datasets suggested an upper bound of $r = 0.4$. In theory, a large proportion of this agreement is accounted for by paradigmatic associations which we are not likely to fully reproduce with these first-order measures. By this standard, the general levels of



Kent

Windowed (w = 2 ... 1250)	2	10	50	250	1250	Best
PMI	0.150	0.193	0.186	0.170	0.143	0.193
LL	0.152	0.200	0.199	0.187	0.163	0.200
CP	0.153	0.203	0.203	0.189	0.169	0.203
CP_{sig}	0.152	0.196	0.188	0.172	0.154	0.196
SCI	0.152	0.204	**0.210**	0.206	0.182	**0.210**
SCI_{sig}	0.154	0.205	0.206	0.192	0.167	0.206
PMI_{sig}	0.152	0.199	0.201	0.193	0.166	0.201
Best	0.154	0.205	**0.210**	0.206	0.182	

Windowless (m = -2 ... 2)	-2	-1	0	1	2	BEST
CD	0.187	0.175	0.131	0.097	0.082	0.187
NWCD	0.201	0.201	0.181	0.158	0.147	0.201
CWCD	0.229	**0.235**	0.227	0.213	0.206	**0.235**
$NWCD_{sig}$	0.156	0.167	0.178	0.181	0.186	0.186
$CWCD_{sig}$	0.170	0.191	0.214	0.212	0.208	0.214
Best	0.229	**0.235**	0.227	0.213	0.208	

Difference	
Baseline	0.113
Upper	0.4
Wind'd	0.210
Wind'less	0.235
▲ Acc	**24.6%**
▼ Err	**1.2%**
Sig	p<0.05

Difference (1m)	
Baseline	0.09
Upper	0.4
Wind'd	0.141
Wind'less	0.143
▲ Acc	**3.5%**
▼ Err	**0.1%**
Sig	p>0.4

Minnesota

Windowed (w = 2 ... 1250)	2	10	50	250	1250	Best
PMI	0.165	0.208	0.197	0.186	0.142	0.208
LL	0.164	0.210	0.202	0.189	0.153	0.210
CP	0.162	0.209	0.199	0.183	0.151	0.209
CP_{sig}	0.161	0.202	0.187	0.169	0.142	0.202
SCI	0.165	0.214	0.211	0.204	0.165	0.214
SCI_{sig}	0.166	**0.215**	0.208	0.192	0.157	**0.215**
PMI_{sig}	0.167	0.214	0.210	0.202	0.163	0.214
Best	0.167	**0.215**	0.211	0.204	0.165	

Windowless (m = -2 ... 2)	-2	-1	0	1	2	BEST
CD	0.199	0.181	0.125	0.083	0.066	0.199
NWCD	0.219	0.218	0.195	0.167	0.154	0.219
CWCD	0.236	**0.239**	0.219	0.197	0.188	**0.239**
$NWCD_{sig}$	0.169	0.180	0.188	0.187	0.190	0.190
$CWCD_{sig}$	0.174	0.192	0.204	0.194	0.189	0.204
Best	0.236	**0.239**	0.219	0.197	0.190	

Difference	
Baseline	0.091
Upper	0.4
Wind'd	0.215
Wind'less	0.239
▲ Acc	**23.2%**
▼ Err	**1.1%**
Sig	p<0.05

Difference (1m)	
Baseline	0.08
Upper	0.4
Wind'd	0.141
Wind'less	0.154
▲ Acc	**20.5%**
▼ Err	**0.4%**
Sig	p>0.1

Edinburgh

Windowed (w = 2 ... 1250)	2	10	50	250	1250	Best
PMI	0.081	0.095	0.093	0.084	0.053	0.095
LL	0.081	0.095	0.092	0.082	0.052	0.095
CP	0.081	0.096	0.096	0.089	0.063	0.096
CP_{sig}	0.079	0.089	0.083	0.069	0.043	0.089
SCI	0.082	0.099	**0.103**	0.099	0.071	**0.103**
SCI_{sig}	0.082	0.096	0.092	0.077	0.046	0.096
PMI_{sig}	0.081	0.094	0.088	0.072	0.037	0.094
Best	0.082	0.099	**0.103**	0.099	0.071	

Windowless (m = -2 ... 2)	-2	-1	0	1	2	BEST
CD	0.073	0.074	0.063	0.047	0.041	0.074
NWCD	0.091	0.097	0.098	0.088	0.083	0.098
CWCD	0.111	0.119	**0.122**	0.115	0.111	**0.122**
$NWCD_{sig}$	0.026	0.032	0.040	0.043	0.046	0.046
$CWCD_{sig}$	0.043	0.055	0.077	0.090	0.097	0.097
Best	0.111	0.119	**0.122**	0.115	0.111	

Difference	
Baseline	0.059
Upper	N/A
Wind'd	0.103
Wind'less	0.122
▲ Acc	**66.8%**
▼ Err	**0.6%**
Sig	p<0.0001

Difference (1m)	
Baseline	0.05
Upper	N/A
Wind'd	0.076
Wind'less	0.108
▲ Acc	**98.9%**
▼ Err	**3.5%**
Sig	p<0.0001

Florida

Windowed (w = 2 ... 1250)	2	10	50	250	1250	Best
PMI	0.114	0.148	0.157	0.141	0.112	0.157
LL	0.114	0.153	0.160	0.140	0.092	0.160
CP	0.114	0.153	0.161	0.141	0.109	0.161
CP_{sig}	0.113	0.149	0.150	0.127	0.098	0.150
SCI	0.114	0.154	**0.167**	0.153	0.121	**0.167**
SCI_{sig}	0.115	0.154	0.163	0.143	0.110	0.163
PMI_{sig}	0.114	0.150	0.159	0.142	0.111	0.159
Best	0.115	0.154	**0.167**	0.153	0.121	

Windowless (m = -2 ... 2)	-2	-1	0	1	2	BEST
CD	0.135	0.130	0.109	0.085	0.075	0.135
NWCD	0.159	0.161	0.156	0.142	0.134	0.161
CWCD	0.171	**0.174**	0.169	0.156	0.149	**0.174**
$NWCD_{sig}$	0.101	0.109	0.120	0.123	0.125	0.125
$CWCD_{sig}$	0.103	0.115	0.133	0.137	0.139	0.139
Best	0.171	**0.174**	0.169	0.156	0.149	

Difference	
Baseline	0.049
Upper	N/A
Wind'd	0.167
Wind'less	0.174
▲ Acc	**8.7%**
▼ Err	**0.2%**
Sig	p<0.1

Difference (1m)	
Baseline	0.04
Upper	N/A
Wind'd	0.110
Wind'less	0.109
▲ Acc	**-2.9%**
▼ Err	**0.0%**
Sig	p>0.4

Figure 1: Correlations for window-based and windowless measures on a 10 million word corpus

correlation seen here (for these datasets $r = 0.235$ and $r = 0.239$ respectively) seem very reasonable.

What is immediately clear from Figure 1 is that, for the range of parameters tested here, we see a relatively small but statistically significant improvement across four of the five datasets when adopting the distance-based approach.

The correlations are unsurprisingly lower across the board for the much smaller 1 million word corpus. Here, the best distance-based measure statistically significantly outperforms the best window-based one (with a significance level of $p < 0.0001$) on one out of four datasets, while the differences are not great enough to be considered statistically significant on the other three datasets. There is therefore some evidence that the benefits observed with the larger corpus hold in the presence of limited data, which is in support of the general theory that distance-based methods capture more information from the corpus at the co-occurrence level (Washtell, 2009). It remains clear, however, that no method is presently a substitute for using a larger corpus.

In terms of optimum configurations, we find that for the frequency-based approach with the larger corpus, a window size of around +/-10 to +/-50 words more or less consistently produces the best results, irrespective of association the measure. Interestingly on the small corpus the tendency appears to be towards a somewhat larger window size than with the larger corpus. This may be related to the larger windows' increased resilience to data-sparseness. Somewhat surprisingly, we also see that our assymmetric association measures SCI and SCI_{sig} perform the best overall amongst the windowed measures, largely irrespective of the window or corpus, size.

In the large corpus, the best distance-based measure is the asymmetric $CWCD$, with the significance corrected measure $CWCD_{sig}$ showing greater strength in the small corpus: perhaps, again, for its improved reliability in the presence of very low-frequency data. The optimum mean for the distance-based parameterisations is somewhere around $m = -1$ (the harmonic) to $m = 0$ (the geometric). We find this unsurprising as the typical distribution of inter-word distances in a corpus is heavily skewed towards the smaller distances - indeed even a random corpus exhibits this characteristic with the distances following a geometric distribution.

5 Task II: Agreement with strongest human associations

The correlation evalation presented considers all word pairs present in the human datasets. However, human association norms tend to contain a very long tail of *hapax legomena* - responses which were given by only one individual. Such responses are extremely difficult for corpus-based

association measures to predict, and given that there is so little consensus amongst human respondents over these items, it is probably not particularly useful to do so. Rather, it might be most useful to predict common or majority human responses.

5.1 Evaluation measure and Upper Bound

For the strongest human response to each cue in the human datasets, its rank was calculated amongst all $33,000$ possible responses to that cue, according to each association measure and parameterisation. Where there were tied scores for various responses, a median rank was assigned. As a rough upper bound, we would be impressed by a computer system which was able to predict the most popular human response as often as a randomly selected individual in the human experiments happened to chose the most popular response.

5.2 Task II Results

Figure 2 illustrates the range of computational association scores attributed to only the strongest human responses. The position of the strongest human response to each cue word, within the computationally-ranked lists of all possible responses, is plotted on the y-axis. For each association measure the points are ordered from best to worst along the x-axis. In the ideal case therefore, the most popular human response for every cue word would appear at rank 1 amongst the computer-generated responses, resulting in a horizonal line at y=1. Generally speaking therefore, the smaller the area above a line the better the performance of a measure.

Three summary statistics can be derived from Figure 2:

1) The number of most popular human responses that are correctly predicted by a measure is indicated by the x-position at which its line departs from y=1. This can be seen to be around 11% for CWCD$_{sig}$ and is zero for the two best PMI parameterizations, with other illustrated measures performing intermediately.

2) The width of the flat horizontal tails at the opposite corner of the figure indicate the proportion of the cue words for which a measure was unable to differentiate the strongest human response from the large contingent of zero association scores resulting from unobservable co-occurrences. This tail is non-existent for CWCD$_{sig}$, but afflicts some 25% and 62% of cue words under the two best PMI parameterizations, again with other illustrated measures performing intermediately.

3) The median rank of the most popular human response for each measure can be read of on the y-axis at the horizontal mid-point (indicated by a feint vertical line).

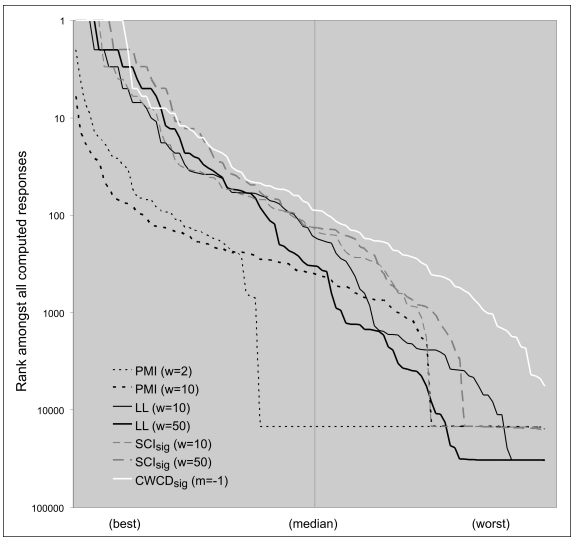


Figure 2: Agreement of computational measures with strongest human responses

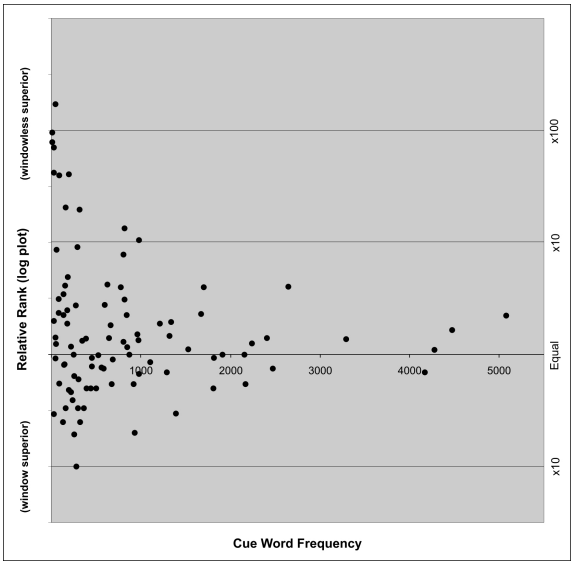


Figure 3: Relative agreement of computational measures with strongest human responses

The results shown are for the Kent dataset, and are highly typical. Included in the figure are the three frequency-based configurations with the highest median rank: SCI$_{sig}$ at window sizes $w = 10$ and $w = 50$, and standard LL at $w = 10$. Three

other frequency-based configurations are included for contrast. Also included is the single windowless configuration with the highest median rank - in this case CWCD$_{sig}$ using the harmonic mean. Several other windowless configurations (notably CWCD and the nearby means) and had very similar profiles.

Figure 3 shows the magnitude of the difference in the ranking of each of the same 100 strong human cue/response pairs, between the best windowless versus best windowed method. Points above the axis represent those cue/response pairs which the windowless method ranked more highly, and vice-versa. The points have been ordered on the x-axis according the the cue word frequency.

5.3 Discussion

Noteworthy, studying Figure 2, is the great sensitivity of the frequency-based measures to the window size parameter. There exists a cut-off point, linked to window size, beyond which the frequency-based measures are unable to make any differentiation between the desired human response and a large portion of the $33,000$ candidate responses. This is almost certainly due to a lack of evidence in the presence of very low frequency words. Log-Likelihood performs somewhat better in this respect, as it takes negative information into account.

Although the distance-based approach follows the same general trend as the other measures, it is nonetheless able to generate a distinct non-zero association score for *every* strong human response and overall it aptly ranks them more highly. A larger number these responses are actually ranked first (i.e. successfully predicted) by the distance-based approach. In fact this number is comparable to, and sometimes exceeds, the upper-bound of 10% implied by taking the average proportion of human respondents who give the most popular response to a given cue.

Whilst Figure 2 showed that overall the windowless method fairs better, on a per-cue basis (Figure 3) things are a little more interesting: For a little over a third of cue-words the windowed method actually appears to perform somewhat better. For the majority however, the windowless approach performs *considerably better* (note that the y-axis scale is logarithmic). It can also be seen that the difference between the methods is most pronounced for low frequency cue words, with responses to some cues exhibiting a relative ranking of around one-hundred times lower for the windowed method. This further supports the theory that the windowless methods are better able to exploit sparse data.

6 Conclusions and Future work

This paper presented the first empirical comparison of window-based and the relatively recently introduced windowless association measures, using their ability to reproduce human association scores as a testbed. We show that the best windowless measures are always at least as good as the best window-based measures, both when it comes to overall correlation with human association scores and predicting the strongest human response. In addition, for several human association sets, they perform significantly better. Although not all parameter settings and corpus sizes could be explored, we conclude that it is worthwhile investigating windowless association measures further. As a side-benefit, we have also introduced new variants of existing frequency-based association measures and shown them to perform as well as or better than their existing counterparts. Although these measures were semi-principled in their construction, a deeper understanding of why they work so well is needed. This may in turn lead to the construction of superior windowless measures.

In our own future work, we are especially interested in using higher-order windowless association measures for retrieving paradigmatic relations as well as exploring their use in various NLP applications.

7 Acknowledgements

We would like to extend sincere thanks to Reinhard Rapp for providing us with the Minnesota dataset in digital form, and additional thanks to Eric Atwell for his support.

References

M. Baroni, S. Evert, and A. Lenci, editors. 2008. *Esslli Workshop on Distributional Lexical Semantics*.

L. Burnard, 1995. *Users' Reference Guide, British National Corpus*. British National Corpus Consortium, Oxford, England.

K. Church and P. Hanks. 1989. Word association norms, mutual information, and lexicography. In *Proc. of ACL-89*, pages 76–83.

K. Church and P. Hanks. 1991. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16(1):22–29.

P. Clark and F.C. Evans. 1954. Distance to nearest neighbor as a measure of spatial relationships in populations. *Ecology*, 35:445–453.

J. Curran. 2003. *From distributional to semantic similarity*. Ph.D. thesis, University of Edinburgh.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19:61–74.

S. Evert. 2005. *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. Ph.D. thesis, Institut fr maschinelle Sprachverarbeitung, University of Stuttgart.

D. Hardcastle. 2005. Using the distributional hypothesis to derive coocurrence scores from the British National Corpus. In *Proc. of Corpus Linguistics*.

J. Jenkins. 1970. The 1952 Minnesota word association norms. In L. Postman and G. Keppel, editors, *Norms of word associations*, pages 1–38. Academic press.

G. Kent and A. Rosanoff. 1910. A study of association in insanity. *Amer. J. of Insanity*, pages 317–390.

G. Kiss, C. Armstrong, R. Milroy, and J. Piper. 1973. An associative thesaurus of English and its computer analysis. In A. Aitken, R. Bailey, and N. Hamilton-Smith, editors, *The Computer and Literary Studies*. Edinburgh University Press.

B. Krenn and S. Evert. 2001. Cam we do better than frequency? a case study on extracting pp-verb collocations. In *Proc. of the ACL Workshop on Collocations*.

A. Lamjiri, O. El Demerdash, and L. Kosseim. 2004. Simple features for statistical word sense disambiguation. In *Proc. of SENSEVAL-2004*.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *Proc. of COLING-ACL-98*.

Lukas Michelbacher, Stefan Evert, and Hinrich Schütze. 2007. Asymmetric association measures. In *Proc. of RANLP-2007*.

D. Nelson, C. McEvoy, J. Walling, and J. Wheeler. 1980. The University of South Florida homograph norms. *Behaviour Research Methods and Instrumentation*, 12:16–37.

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

T. Pedersen, S. Patwardhan, and J. Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *Proc. of the 21st National Conference on Artificial Intelligence; 2004*.

R. Rapp. 2002. The computation of word associations: comparing syntagmatic and paradigmatic approaches. In *Proc of COLING 2002*.

D.L. Sackett. 2001. Why randomized controlled trials fail but needn't: 2. failure to employ physiological statistics, or the only formula a clinician-trialist is every likely to need (or understand). *Canadian Medical Association Journal*, 165(9):1226–1237.

P. Savicki and J. Hlavacova. 2002. Measures of word commonness. *Journal of Quantitative Linguistcs*, 9(3):215–231.

E. Terra and C. Clarke. 2004. Fast computation of lexical affinity models. In *Proc of COLING 2004*.

Renata Vieira and Massimo Poesio. 2000. An empirically-based system for processing definite descriptions. *Computational Linguistics*, 26(4), December.

X. Wang, 2005. *Robust Utilization of Context in Word Sense Disambiguation*, chapter Modeling and Using Context, pages 529–541. Springer Lecture Notes in Computer Science.

J. Washtell. 2009. Co-dispersion: A windowless approach to lexical association. In *Proc. of EACL-2009*.

M. Wettler and R. Rapp. 1993. Computation of word associations based on the co-ocurrences of words in large corpora. In *Proc. of the First Workshop on Very Large Corpora*.

K. White and L. Abrams. 2004. Free associations and dominance ratings of homophones for young and older adults. *Behaviour Research Methods, Instruments and Computers*, 36:408–420.

D. Yarowsky and R Florian. 2002. Evaluating sense disambiguation across diverse parameter spaces. *Natural Language Engineering*, 8(4):293–310.

Improving Verb Clustering with Automatically Acquired Selectional Preferences

Lin Sun and Anna Korhonen
University of Cambridge, Computer Laboratory
15 JJ Thomson Avenue, Cambridge CB3 0GD, UK
ls418,alk23@cl.cam.ac.uk

Abstract

In previous research in automatic verb classification, syntactic features have proved the most useful features, although manual classifications rely heavily on semantic features. We show, in contrast with previous work, that considerable additional improvement can be obtained by using semantic features in automatic classification: verb selectional preferences acquired from corpus data using a fully unsupervised method. We report these promising results using a new framework for verb clustering which incorporates a recent subcategorization acquisition system, rich syntactic-semantic feature sets, and a variation of spectral clustering which performs particularly well in high dimensional feature space.

1 Introduction

Verb classifications have attracted a great deal of interest in natural language processing (NLP). They have proved useful for various important NLP tasks and applications, including e.g. parsing, word sense disambiguation, semantic role labeling, information extraction, question-answering, and machine translation (Swier and Stevenson, 2004; Dang, 2004; Shi and Mihalcea, 2005; Zapirain et al., 2008).

Verb classes are useful because they offer a powerful tool for generalization and abstraction which can be beneficial when faced e.g. with the problem of data sparsity. Particularly useful can be classes which capture generalizations over a range of (cross-)linguistic properties, such as the ones proposed by Levin (1993). Being defined in terms of similar meaning and (morpho-)syntactic behaviour of words, Levin style classes generally incorporate a wider range of properties than e.g. classes defined solely on semantic grounds (Miller, 1995).

In recent years, a variety of approaches have been proposed for automatic induction of verb classes from corpus data (Schulte im Walde, 2006; Joanis et al., 2008; Sun et al., 2008; Li and Brew, 2008; Korhonen et al., 2008; Ó Séaghdha and Copestake, 2008; Vlachos et al., 2009). This work opens up the opportunity of learning and tuning classifications tailored to the application and domain in question. Although manual classification may always yields higher accuracy, automatic verb classification is cost-effective and gathers statistical information as a side-effect of the acquisition process which is difficult for humans to gather but can be highly useful for NLP applications.

To date, both supervised and unsupervised machine learning (ML) methods have been proposed for verb classification and used to classify a variety of features extracted from raw, tagged and/or parsed corpus data. The best performing features on cross-domain verb classification have been syntactic in nature (e.g. syntactic slots, subcategorization frames (SCFs)). Disappointingly, semantic features have not yielded significant additional improvement, although they play a key role in manual and theoretical work on verb classification and could thus be expected to offer a considerable contribution to classification performance.

Since the accuracy of automatic verb classification shows room for improvement, we further investigate the potential of semantic features – verb selectional preferences (SPs) – for the task. We introduce a novel approach to verb clustering which involves the use of (i) a recent subcategorization frame (SCF) acquisition system (Preiss et al., 2007) which produces rich lexical, SCF and syntactic data, (ii) novel syntactic-semantic feature sets extracted from this data which incorporate a variety of linguistic information, including SPs, and (iii) a new variation of spectral cluster-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 638–647,
Singapore, 6-7 August 2009.

ing based on the MNCut algorithm (Meila and Shi, 2001) which is well-suited for dealing with the resulting, high dimensional feature space.

Using this approach, we show on two well-established test sets that automatically acquired SPs can be highly useful for verb clustering. They yield high performance when used in combination with syntactic features. We obtain our promising results using a fully unsupervised approach to SP acquisition which differs from previous approaches in that it does not exploit WordNet (Miller, 1995) or other lexical resources. It is based on clustering argument head data in the grammatical relations associated with verbs.

We describe our features in section 2 and the clustering methods in section 3. Experimental evaluation and results are reported in sections 4 and 5, respectively. Section 6 provides discussion and describes related work, and section 7 concludes.

2 Features

Our target classification is the taxonomy of Levin (1993) where verbs taking similar diathesis alternations are assumed to share meaning components and are organized into semantically coherent classes. The main feature of this classification is a diathesis alternation which manifests at the level of syntax in alternating sets of SCF (e.g. in the causative/inchoative alternation an NP frame alternates with an intransitive frame: *Tony broke the window* ↔ *The window broke*).

Since automatic detection of diathesis alternations is very challenging (McCarthy, 2001), most work on automatic classification has exploited the fact that similar alternations tend to result in similar SCFs. The research reported so far[1] has used mainly syntactic features for classification, ranging from shallow syntactic slots (e.g. NPs preceding or following the verb) to SCFs. Some researchers have discovered that supplementing basic syntactic features with information about adjuncts, co-occurrences, tense, and/or voice of the verb have resulted in better performance.

However, additional information about semantic SPs of verbs has not yielded considerable improvement on verb classification although SPs can be strong indicators of diathesis alternations (McCarthy, 2001) and although fairly precise semantic descriptions, including information about verb selectional restrictions, can be assigned to the majority of Levin classes, as demonstrated by VerbNet (Kipper-Schuler, 2005).

SP acquisition from undisambiguated corpus data is arguably challenging (Brockmann and Lapata, 2003; Erk, 2007; Bergsma et al., 2008). It is especially challenging in the context of verb classification where SP models are needed for specific syntactic slots for which the data may be sparse, and the resulting feature vectors integrating both syntactic and semantic features may be high dimensional. However, we wanted to investigate whether better results could be obtained if the features were optimised for richness, the feature extraction for accuracy, and a clustering method capable of dealing with the resulting high dimensional feature space was employed.

2.1 Feature extraction

We adopted a recent SCF acquisition system which has proved more accurate than previous comparable systems[2] but which has not been employed for verb clustering before: the system of Preiss et al. (2007). This system tags, lemmatizes and parses corpus data using the current version of the RASP (Robust Accurate Statistical Parsing) toolkit (Briscoe et al., 2006), and on the basis of resulting grammatical relations (GRs) assigns each occurrence of a verb to one of 168 verbal SCFs classes[3].

The system provides a filter which can be used to remove adjuncts from the resulting lexicon. We do not employ this filter since adjuncts have proved informative for verb classification (Sun et al., 2008; Joanis et al., 2008). However, we do frequency-based thresholding to minimise the noise (e.g. erroneous scfs) and sparse data in verb classification and to ensure that only features supported by several verbs are used in classification: we only consider SCFs and GRs which have frequency larger than 40 with 5 or more verbs[4].

The system produces a rich lexicon which includes raw and processed input sentences and provides a variety of material for verb clustering, including e.g. (statistical) information related to the part-of-speech (POS) tags, GRs, SCFs, argument heads, and adjuncts of verbs. Using this material, we constructed a wide range of feature sets

[1] See section 6 for discussion on previous work.

[2] See Preiss et al. (2007) for the details of evaluation.

[3] We used an implementation of the SCF classifier provided by Paula Buttery.

[4] These and other threshold values mentioned in this paper were determined empirically on corpus data.

for experimentation, both shallow and deep syntactic and semantic features. As described below, some of the feature types have been employed in previous works and some are novel.

2.2 Feature sets

The first feature set F1 includes information about the lexical context (co-occurrences) of verbs which has proved useful for supervised verb classification (Li and Brew, 2008):

F1: Co-occurrence (CO): We adopt the best method of Li and Brew (2008) where collocations are extracted from the four words immediately preceding and following a lemmatized verb. Stop words are removed prior to extraction, and the 600 most frequent resulting COs are kept.

F2-F3 provide information about lexical preferences of verbs in argument head positions of specific GRs associated with the verb:

F2: Prepositional preference (PP): the type and frequency of prepositions in the indirect object relation.

F3: Lexical preference (LP): the type and frequency of nouns and prepositions in the subject, object, and indirect object relation.

All the other feature sets include information about SCFs which have been widely employed in verb classification, e.g. (Schulte im Walde, 2006; Sun et al., 2008; Li and Brew, 2008; Korhonen et al., 2008). F4-F7 include basic SCF information and/or refine it with additional information which has proved useful in previous works:

F4: SCFs and relative frequencies with verbs. SCFs abstract over particles and prepositions.

F5: F4 with COs (F1). The SCF and CO feature vectors are concatenated.

F6: F4 with the tense of the verb. The frequency of verbal POS tags is calculated specific to each SCF.

F7: F4 with PPs (F2). This feature parameterizes SCFs for prepositions.

F8: Basic SCF feature corresponding to F4 but extracted from the VALEX lexicon (Korhonen et al., 2006)[5].

The following 9 feature sets are novel. They build on F7, refining it further. F9-F11 refine F7 with information about LPs:

F9: F7 with F3 (subject only)

F10: F7 with F3 (object only)

F11: F7 with F3 (subject, object, indirect object)

F12-17 refine F7 with SPs. We adopt a fully unsupervised approach to SP acquisition. We acquire the SPs by

1. taking the GR relations (subject, object, indirect object) associated with verbs,

2. extracting all the argument heads in these relations which occur with frequency > 20 with more than 3 verbs, and

3. clustering the resulting N most frequent argument heads into M classes using the spectral clustering method described in the following section.

We tried the N settings $\{200, 500\}$ and the M settings $\{10, 20, 30, 80\}$. The best settings $N = 200, M = 20$ and $N = 500, M = 30$ are reported in this paper. We enforce the features to be shared by all the potential members of a verb class. The expected class size is approximately N/K, and we allow for 10% outliers (the features occurring less than $(N/K) \times 0.9$ verbs are thus removed).

The resulting SPs are combined with SCFs in a similar fashion as LPs are combined with SCFs in F9-F11:

F12-F14: as F9-F11 but SPs (20 clusters from 200 argument heads) are used instead of LPs

F15-F17: as F9-F11 but SPs (30 clusters from 500 argument heads) are used instead of LPs

[5]This feature was included to enable comparing the contribution of the recent SCF system to that of an older, comparable system which was used for constructing the VALEX lexicon.

3 Clustering methods

We use two clustering methods: (i) pairwise clustering (PC) which obtained the best performance in comparison with several other methods in recent work on biomedical verb clustering (Korhonen et al., 2008), and (ii) a method which is new to the task (and to the best of our knowledge, to NLP): a variation of spectral clustering which exploits the MNCut algorithm (Meila and Shi, 2001) (SPEC). Spectral clustering has been shown to be effective for high dimensional and non-convex data in NLP (Chen et al., 2006) and it has been applied to German verb clustering by Brew and Schulte im Walde (2002). However, previous work has used Ng et al. (2002)'s algorithm, while we adopt the MNCut algorithm. The latter has shown a wider applicability (von Luxburg, 2007; Verma and Meila, 2003) and it can be justified from the random walk view, which has a clear probabilistic interpretation.

Clustering groups a given set of items (verbs in our experiment) $V = \{v_n\}_{n=1}^{N}$ into a disjoint partition of K classes $I = \{I_k\}_{k=1}^{K}$. Both our algorithms take a similarity matrix as input. We construct this from the skew divergence (Lee, 2001). The skew divergence between two feature vectors v and v' is $d_{skew}(v, v') = D(v'||a \cdot v + (1-a) \cdot v')$ where D is the KL-divergence. v is smoothed with v'. The level of smoothing is controlled by a whose value is set to a value close to 1 (e.g. 0.9999). We symmetrize the skew divergence as follows: $d(v, v')_{sskew} = \frac{1}{2}(d_{skew}(v, v') + d_{skew}(v', v))$.

SPEC is typically used with the Radial Basis Function (RBF) kernel. We adopt a new kernel similar to the symmetrized KL divergence kernel (Moreno et al., 2004) which avoids the need for scale parameter estimation.

$$w(v, v') = \exp(-d_{sskew}(v, v'))$$

The similarity matrix W is constructed where $W_{ij} = w(v_i, v_j)$.

Pairwise clustering

PC (Puzicha et al., 2000) is a method where a cost criterion guides the search for a suitable partition. This criterion is realized through a cost function of the similarity matrix W and partition I:

$$H = -\sum n_j \cdot \text{Avgsim}_j,$$
$$\text{Avgsim}_j = \frac{\sum_{\{a,b \in A_j\}} w(a,b)}{n_j \cdot (n_j - 1)}$$

where n_j is the size of the j_{th} cluster and Avgsim_j is the average similarity between cluster members.

Spectral clustering

In SPEC, the similarities W_{ij} are viewed as the weight on the edges ij of a graph G over V. The similarity matrix W is thus the adjacency matrix for G. The degree of a vertex i is $d_i = \sum_{j=1}^{N} w_{ij}$. A cut between two partitions A and A' is defined to be $\text{Cut}(A, A') = \sum_{m \in A, n \in A'} W_{mn}$.

In MNCut algorithm, the similarity matrix W is transformed to a stochastic matrix P.

$$P = D^{-1}W \quad (1)$$

The degree matrix D is a diagonal matrix where $D_{ii} = d_i$.

It was shown by Meila and Shi (2001) that if P has the K leading eigenvectors that are piecewise constant[6] with respect to a partition I^* and their eigenvalues are not zero, then I^* minimizes the multiway normalized cut(MNCut):

$$\text{MNCut}(I) = K - \sum_{k=1}^{K} \frac{\text{Cut}(I_k, I_k)}{\text{Cut}(I_k, I)}$$

P_{mn} can be interpreted as the transition probability between vertices m, n. The criterion can thus be expressed as $\text{MNCut}(I) = \sum_{k=1}^{K}(1 - P(I_k \rightarrow I_k|I_k))$ (Meila, 2001), which is the sum of transition probabilities across different clusters. The criterion finds the partition where the random walks are most likely to happen within the same cluster.

In practice, the K leading eigenvectors of P is not piecewise constant. But we can extract the partition by finding the approximately equal elements in the eigenvectors using a clustering algorithm like K-means.

The numerator of MNCut is similar to the cost function of PC. The main differences between the two algorithms are: 1) MNCut takes into account of the cross cluster similarity, while PC does not. 2) PC optimizes the cost function using deterministic annealing, whereas SPEC uses eigensystem decomposition.

The spectral clustering algorithm is based on the Multicut algorithm (Meila and Shi, 2001).

[6] The eigenvector v is piecewise constant with respect to I if $v(i) = v(j) \forall i, j \in I_k$ and $k \in 1, 2...K$

Input: Dataset S, Number of clusters K

1. Compute similarity matrix W and Degree matrix D
2. Construct stochastic matrix P using equation 1
3. Compute the eigenvalues and eigenvectors $\{\lambda_n, x_n\}_{n=1}^{N}$ of P, where $\lambda_n \geq \lambda_{n+1}$, form a matrix $X = [x_2, \ldots, x_k]$ by stacking the eigenvectors in columns.
4. Form a matrix Y from X by normalizing the row sums to have norm 1: $Y_{ij} = X_{ij}/(\sum_j X_{ij}^2)^{\frac{1}{2}}$
5. Consider the row of Y to be the transformed feature vectors for each verb and cluster them into clusters $C_1 \ldots C_k$ using K-means clustering algorithm.

Output: Clusters $C_1 \ldots C_k$

4 Experimental evaluation

4.1 Test sets

We employed two test sets which have been used to evaluate previous work on English verb classification:

T1 The test set of Joanis et al. (2008) provides a classification of 835 verbs into 15 (some coarse, some fine-grained) Levin classes. 11 tests are provided for 2-14 way classifications. We employ the 14 way classification because this corresponds the closest to our target (Levin's fine-grained) classification[7]. We select 586 verbs according to Joanis et al.'s selection criteria, resulting in 10-120 verbs per class. We restrict the class imbalance to 1:1.5.[8]. This yields 205 verbs (10-15 verbs per class) which is similar to the sub-set of T1 employed by Stevenson and Joanis (2003).

T2 The test set of Sun et al. (2008) classifies 204 verbs to 17 fine-grained Levin classes, so that each class has 12 member verbs.

Table 1 shows the classes in T1 and T2.

4.2 Data processing

For each verb in T1 and T2, we extracted all the occurrences (up to 10,000) from the raw corpus data gathered originally for constructing the VALEX lexicon (Korhonen et al., 2006). The data was gathered from five corpora, including e.g. the British National Corpus (Leech, 1992) and the North American News Text Corpus (Graff, 1995). The average frequency of verbs in T1 was 1448 and T2 2166, showing that T1 is a more sparse data set.

The data was first processed using the feature extraction module. Table 2 shows (i) the total number of features in each feature set and (ii) the average per verb in the resulting lexicons for T1 and T2.

We normalized the feature vectors by the sum of the feature values before applying the clustering techniques. Since both clustering algorithms have

[7]However, the correspondence is not perfect with half of the classes including two or more Levin's fine-grained classes.

[8]Otherwise, in the case of a large class imbalance the evaluation measure would be dominated by the classes with large population.

T1		T2	
Object Drop	26.{1,3,7}	Remove	10.1
Recipient	13.{1,3}	Send	11.1
Admire	31.2	Get	13.5.1
Amuse	31.1	Hit	18.1
Run	51.3.2	Amalgamate	22.2
Sound	43.2	Characterize	29.2
Light & Substance	43.{1,4}	Peer	30.3
		Amuse	31.1
Cheat	10.6	Correspond	36.1
Steal & Remove	10.{5,1}	Manner of speaking	37.3
Wipe	10.4.{1,2}	Say	37.7
Spray / Load	9.7	Nonverbal expression	40.2
Fill	9.8		
Putting	9.1-6	Light	43.1
Change of State	45.1-4	Other change of state	45.4
		Mode with Motion	47.3
		Run	51.3.2
		Put	9.1

Table 1: Levin classes in T1 and T2

		T1		T2	
		total	avg	total	avg
CO	F1	1328	764	743	382
LP (p)	F2	61	37	55	25
LP (all)	F3	2521	526	1481	295
SCF	F4	88	46	86	38
SCF+CO	F5	1466	833	856	422
SCF+POS	F6	319	114	299	87
SCF+P	F7	282	96	273	76
SCF (V)	F8	-	-	92	45
SCF+LP (s)	F9	1747	324	1474	225
SCF+LP (o)	F10	2817	424	2319	279
SCF+LP (all)	F11	4250	649	3515	426
SCF+SP20 (s)	F12	821	235	690	145
SCF+SP20 (o)	F13	792	218	706	135
SCF+SP20 (all)	F14	1333	357	1200	231
SCF+SP30 (s)	F15	977	274	903	202
SCF+SP30 (o)	F16	1026	273	1012	205
SCF+SP30 (all)	F17	1720	451	1640	330

Table 2: (i) The total number of features and (ii) the average per verb for all the feature sets

an element of randomness, we run them multiple times. The step 5 of SPEC (K-means) was run for 50 times. The result that minimizes the distortion (the distances to cluster centroid) is reported. PC was run 20 times, and the results are averaged.

4.3 Evaluation measures

To facilitate meaningful comparisons, we employ the same measures for evaluation as previously employed e.g. by Korhonen et al. (2008); Ó Séaghdha and Copestake (2008).

The first measure is modified purity (mPUR) – a global measure which evaluates the mean precision of clusters. Each cluster is associated with its prevalent class. The number of verbs in a cluster K that take this class is denoted by $n_{prevalent}$(K). Verbs that do not take it are considered as errors. Clusters where $n_{prevalent}$(K) = 1 are disregarded as not to introduce a bias towards singletons:

$$\text{mPUR} = \frac{\sum_{n_{prevalent(k_i)>2}} n_{prevalent(k_i)}}{\text{number of verbs}}$$

The second measure is weighted class accuracy (ACC): the proportion of members of dominant clusters DOM-CLUST$_i$ within all classes c_i.

$$\text{ACC} = \frac{\sum_{i=1}^{C} \text{verbs in DOM-CLUST}_i}{\text{number of verbs}}$$

mPUR and ACC can be seen as a measure of precision(P) and recall(R) respectively. We calculate F measure as the harmonic mean of P and R:

$$\text{F} = \frac{2 \cdot \text{mPUR} \cdot \text{ACC}}{\text{mPUR} + \text{ACC}}$$

The random baseline(BL) is calculated as follows:

$$\text{BL} = 1/\text{number of classes}$$

5 Results

5.1 Quantitative evaluation

Table 3 includes the F-measure results for all the feature sets when the two methods (PC and SPEC) are used to cluster verbs in the test sets T1 and T2, respectively. A number of tendencies can be observed in the results. Firstly, the results for T2 are clearly better than those for T1. Including a higher number of verbs lower in frequency from classes of variable granularity, T1 is probably a more challenging test set than T2. T2 is controlled for the number and frequency of verbs to facilitate cross-class comparisons. While this may contribute to better results, T2 is a more accurate test set for us in the sense that it offers a better correspondence with our target (fine-grained Levin) classes.

		T1		T2	
		PC	SPEC	PC	SPEC
	BL	7.14	7.14	5.88	5.88
CO	F1	15.62	33.85	17.86	40.94
LP (p)	F2	40.40	38.97	50.98	49.02
LP (all)	F3	42.94	47.50	41.08	74.55
SCF	F4	34.22	36.16	52.33	57.78
SCF+CO	F5	26.43	28.70	19.52	29.10
SCF+POS	F6	36.14	34.75	44.44	46.70
SCF+P	F7	43.57	43.85	63.40	63.28
SCF (V)	F8	-	-	34.08	38.30
SCF+LP (s)	F9	47.72	56.09	65.94	71.65
SCF+LP (o)	F10	43.09	48.43	57.11	73.97
SCF+LP (all)	F11	45.87	54.63	56.30	72.97
SCF+SP20 (s)	F12	46.67	**57.75**	39.52	71.67
SCF+SP20 (o)	F13	44.95	51.70	40.76	70.78
SCF+SP20(all)	F14	48.19	55.12	39.68	73.09
SCF+SP30 (s)	F15	45.89	56.10	64.44	**80.35**
SCF+SP30 (o)	F16	42.01	48.74	52.75	70.52
SCF+SP30(all)	F17	46.66	52.68	51.07	68.67

Table 3: Results on testsets T1 and T2

Secondly, the difference between the two clustering methods is clear: the new SPEC outperforms PC on both test sets and across all the feature sets. The performance of the two methods is still fairly similar with the more basic, less sparse feature sets (F1-F2, F4, F6-7) but when the more sophisticated feature sets are used (F3, F5, F9-F17) SPEC performs considerably better. This demonstrates that it is clearly a better suited method for high dimensional feature sets.

Comparing the feature sets, the simple co-occurrence based F1 performs clearly better than the random baseline. F2 and F3 which exploit lexical data in the argument head positions of GRs prove significantly better than F1. F3 yields surprisingly good results on T2: it is the second best feature set on this test set. Also on T1, F3 performs better than the SCF-based feature sets F4-F7. This demonstrates the usefulness of lexical data when obtained from argument positions in relevant GRs.

Our basic SCF feature set F4 performs considerably better than the comparable feature set F8 obtained from the VALEX lexicon. The difference is 19.50 in F-measure. As both lexicons were extracted from the same corpus data, the improvement can be attributed to improved parser and SCF acquisition performance (Preiss et al., 2007).

F5-F7 refine the basic SCF feature set F4 further. F5 which combines a SCF with CO information proved the best feature set in the supervised verb classification experiment of Li and Brew (2008). In our experiment, F5 produces substantially lower result than CO and SCF alone (i.e.

F1 and F4). However, our corpus is smaller (Li and Brew used the large Gigaword corpus), our SCFs are different, and our approach is unsupervised, making meaningful comparisons difficult.

F6 combines F4 with information about verb tense. This was not helpful: F6 produces worse results than F4. F7, on the other hand, yields better results than F4 on both test sets. This demonstrates what the previous research has shown: SCF perform better when parameterized for prepositions.

Looking at our novel feature sets F9-F17, F9-F11 combine the most accurate SCF feature set F4 with the LP-based features F2-F3. Although the feature space gets more sparse, all the feature sets outperform F2-F3 on T1. On T2, F3 performs exceptionally well, and thus yields a better result than F9-F11, but F9-F11 nevertheless perform clearly better than the best SCF-based feature set F4 alone. The differences among F9, F10 and F11 are small on T2, but on T1 F9 yields the best performance. It could be that F9 works the best for the more sparse T1 because it suffers the least from data sparsity (it uses LPs only for the subject relation).

F12-F17 replace the LPs in F9-F11 by semantic SPs. When only 20 clusters are used as SP models and acquired from the smaller sample of (200) argument heads (F12-F14), SPs do not perform better than LPs on T2. A small improvement can be observed on T1, especially with F12 which uses only the subject data (yielding the best F measure on T1: 57.75%). However, when 30 more fine-grained clusters are acquired from a bigger sample of (500) argument heads (F15-F17), lower results can be seen on T1. On T2, on the other hand, F15 yields dramatic improvement and we get the best performance for this test set: 80.35% F-measure.

The fact that no improvement is observed when using F16 and F17 on T2 could be explained by the fact that SPs are stronger for the subject position which also suffers less from the sparse data problem than e.g. i. object position. The fact that no improvement is observed on T1 is likely to be due to the fact that verbs have strong SPs only at the finer-grained level of Levin classification. Recall that in T1, as many as half of the classes are coarser-grained.

5.2 Qualitative evaluation

The best performing feature sets on both T1 and T2 were thus our new SP-based feature sets. We conducted qualitative analysis of the best 30 SP clusters in the T2 data created using SPEC to find out whether these clusters were really semantic in nature, i.e. captured semantically meaningful preferences. As no gold standard specific to our verb classification task was available, we did manual cluster analysis using VerbNet (VN) as aid. In VN, Levin classes are assigned with semantic descriptions: the arguments of SCFs involved in diathesis alternations are labeled with thematic roles some of which are labeled with selectional restrictions.

Human	mother, wife, parent, girl, child
Role	patient, student, user, worker, teacher
Body-part	neck, shoulder, back, knee, corner
Authority	committee, police, court, council, board
Organization	society, firm, union, bank, institution
Money	cash, currency, pound, dollar, fund
Amount	proportion, value, size, speed, degree
Time	minute, moment, night, hour, year
Path	street, track, road, stair, route
Building	office, shop, hotel, hospital, house
Region	site, field, area, land, island
Technology	system, model, facility, engine, machine
Task	operation, test, study, analysis, duty
Arrangement	agreement, policy, term, rule, procedure
Matter	aspect, subject, issue, question, case
Problem	difficulty, challenge, loss, pressure, fear
Idea	argument, concept, idea, theory, belief
Power	control, lead, influence, confidence, ability
Form	colour, style, pattern, shape, design
Item	letter, book, goods, flower, card

Table 4: Cluster analysis: 20 clusters, their SP labels, and prototypical member nouns

From the 30 thematic role types in VN, as many as 20 are associated with the 17 Levin classes in T2. The most frequent role in T2 is agent, followed by theme, location, patient, recipient, and source. From the 36 possible selectional restriction types, 7 appear in T2; the most frequent ones being +animate and +organization, followed by +concrete, +location, and +communication.

As SP clusters capture selectional *preferences* rather than *restrictions*, we examined manually whether the 30 clusters (i) capture semantically meaningful classes, and whether they (ii) are plausible given the VN semantic descriptions/restrictions for the classes in T2.

The analysis revealed that all the 30 clusters had a predominant, semantically motivated SP supported by the majority of the member nouns. Although many clusters could be further divided into more specific SPs (and despite the fact that some nouns were clearly misclassified), we were able to assign each cluster a descriptive label characterizing the predominant SP. Table 4 shows 15 sam-

ple clusters, the SP labels assigned to them, and a number of example nouns in these clusters.

When comparing each SP cluster against the VN semantic descriptions/restrictions for T2, we found that each predominant SP was plausible. Also, the SPs frequent in our data were also frequent among the 17 classes according to VN. For example, the many SP clusters labeled as arrangements, issues, ideas and other abstract concepts were also frequent in T2, e.g. among COMMUNICATION (37), CHARACTERISE (29.2), AMALGAMATE (22.2) and other classes.

This analysis showed that the SP models which performed well in verb clustering were semantically meaningful for our task. An independent evaluation using one of the standard datasets available for SP acquisition research (Brockmann and Lapata, 2003) is of course needed to determine how well the acquisition method performs in comparison with other existing methods.

Finally, we evaluated the quality of the verb clusters created using the SP-based features. We found that some of the errors were similar to those seen on T2 when using syntactic features: errors due to polysemy and syntactic idiosyncracy. However, a new error type clearly due to the SP-based feature was detected. A small number of classes got confused because of strong similar SPs in the subject (agent) position. For example, some PEER (30.3) verbs (e.g. *look, peer*) were found in the same cluster with SAY (37.7) verbs (e.g. *shout, yell*) – an error which purely syntactic features do not produce. Such errors were not numerous and could be addressed by developing more balanced SP models across different GRs.

6 Discussion and related work

Although features incorporating semantic information about verb SPs make theoretical sense they have not proved equally promising in previous experiments which have compared them against syntactic features in verb classification. Joanis et al. (2008) incorporated an 'animacy' feature (a kind of a 'SP') which was determined by classifying e.g. pronouns and proper names in data to this single SP class. A small improvement was obtained when this feature was used in conjunction with syntactic features in supervised classification.

Joanis (2002) and Schulte im Walde (2006) experimented with more conventional SPs with syntactic features in English and German verb classification, respectively. They employing top level WordNet (Miller, 1995) and Germanet (Kunze and Lemnitzer, 2002) classes as SP models. Joanis (2002) obtained no improvement over syntactic features, whereas Schulte im Walde (2006) obtained insignificant improvement.

		Method	Result
T1	Li et al. 2008	supervised	66.3
	Joanis et al. 2008	supervised	58.4
	Stevenson et al. 2003	semi-supervised unsupervised	29 31
	SPEC	unsupervised	57.55
T2	Sun et al. 2008	supervised unsupervised	62.50 51.6
	Ó Séaghdha et al. 2008	supervised	67.3
	SPEC	unsupervised	80.35

Table 5: Previous verb classification results

Korhonen et al. (2008) combined SPs with SCFs when clustering biomedical verbs. The SPs were acquired automatically from syntactic slots of SCFs (not from GRs as in our experiment) using PC clustering. A small improvement was obtained using LPs extracted from the same syntactic slots, but the SP clusters offered no improvement. Recently, Schulte im Walde et al. (2008) proposed an interesting SP acquisition method which involves combining EM training and the MDL principle for an verb classification incorporating SPs. However, no comparison against purely syntactic features is provided.

In our experiment, we obtained a considerable improvement over syntactic features, despite using a fully unsupervised approach to both verb clustering and SP acquisition. In addition to the rich, syntactic-semantic feature sets, our good results can be attributed to the clustering technique capable of dealing with them. The potential of spectral clustering for the task was recognised earlier by Brew and Schulte im Walde (2002). Although a different version of the algorithm was employed and applied to German (rather than to English), and although no SP features were used, these earlier experiments did demonstrate the ability of the method to perform well in high dimensional feature space.

To get an idea of how our performance compares with that of related approaches, we examined recent works on verb classification (supervised and unsupervised) which were evaluated on same test sets using comparable evaluation measures. These works are summarized in table 5. ACC and F-measure are shown for T1 and T2, respectively.

On T1, the best performing supervised method reported so far is that of Li and Brew (2008). Li and Brew used Bayesian Multinomial Regression for classification. A range of feature sets integrating COs, SCFs and/or LPs were evaluated. The combination of COs and SCFs gave the best result, shown in the table. Joanis et al. (2008) report the second best supervised result on T1, using Support Vector Machines for classification and features derived from linguistic analysis: syntactic slots, slot overlaps, tense, voice, aspect, and animacy of NPs. Stevenson and Joanis (2003) report a semi- and unsupervised experiment on T1. A feature set similar to that of Joanis et al. (2008) was employed (features were selected in a semi-supervised fashion) and hierarchical clustering was used.

Our unsupervised method SPEC performs substantially better than the unsupervised method of Stevenson et al. and nearly as well as the supervised approach of Joanis et al. (2008) (note, however, that the different experiments involved different sub-sets of T1 so are not entirely comparable).

On T2, the best performing supervised method so far is that of Ó Séaghdha and Copestake (2008) which employs a distributional kernel method to classify SCF features parameterized for prepositions in the automatically acquired VALEX lexicon. Using exactly the same data and feature set, Sun et al. (2008) obtain a slightly lower result when using a supervised method (Gaussian) and a notably lower result when using an unsupervised method (PC clustering). Our method performs considerably better and also outperforms the supervised method of Ó Séaghdha and Copestake (2008).

7 Conclusion and Future Work

We introduced a new approach to verb clustering which involves the use of (i) rich lexical, SCF and GR data produced by a recent SCF system, (ii) novel syntactic-semantic feature sets which combine a variety of linguistic information, and (iii) a new variation of spectral clustering which is particularly suited for dealing with the resulting, high dimensional feature space. Using this approach, we showed on two well-established test sets that automatically acquired SPs can be highly useful for verb clustering. This result contrasts with most previous works but is in line with theoretical work on verb classification which relies not only on syntactic but also on semantic features (Levin, 1993).

In addition to the ideas mentioned earlier, our future plans include looking into optimal ways of acquiring SPs for verb classification. Considerable research has been done on SP acquisition most of which has involved collecting argument headwords from data and generalizing to WordNet classes. Brockmann and Lapata (2003) have showed that WordNet-based approaches do not always outperform simple frequency-based models, and a number of techniques have been recently proposed which may offer ideas for refining our current unsupervised approach (Erk, 2007; Bergsma et al., 2008). The number and type (and combination) of GRs for which SPs can be reliably acquired, especially when the data is sparse, requires also further investigation.

In addition, we plan to investigate other potentially useful features for verb classification (e.g. named entities and preposition classes) and explore semi-automatic ML technology and active learning for guiding the classification. Finally, we plan to conduct a bigger experiment with a larger number of verbs, and conduct evaluation in the context of practical application tasks.

Acknowledgments

Our work was funded by the Dorothy Hodgkin Postgraduate Award, the Royal Society University Research Fellowship, and the EPSRC grant EP/F030061/1, UK. We would like to thank Paula Buttery for letting us use her implementation of the SCF classifier and Yuval Krymolowski for the support he provided for feature extraction.

References

Shane Bergsma, Dekang Lin, and Randy Goebel. Discriminative learning of selectional preference from unlabeled text. In *Proc. of EMNLP*, 2008.

Chris Brew and Sabine Schulte im Walde. Spectral clustering for german verbs. In *Proc. of EMNLP*, 2002.

Ted Briscoe, John Carroll, and Rebecca Watson. The second release of the rasp system. In *Proc. of the COLING/ACL on Interactive presentation sessions*, 2006.

Carsten Brockmann and Mirella Lapata. Evaluating and combining approaches to selectional preference acquisition. In *Proc. of EACL*, 2003.

Jinxiu Chen, Dong-Hong Ji, Chew Lim Tan, and Zheng-Yu Niu. Unsupervised relation disambiguation using spectral clustering. In *Proc. of COLING/ACL*, 2006.

Hoa Trang Dang. *Investigations into the Role of Lexical Semantics in Word Sense Disambiguation*. PhD thesis, CIS, University of Pennsylvania, 2004.

Katrin Erk. A simple, similarity-based model for selectional preferences. In *Proc. of ACL*, 2007.

David Graff. North american news text corpus. *Linguistic Data Consortium*, 1995.

Eric Joanis. Automatic Verb Classification Using a General Feature Space. Master's thesis, University of Toronto, 2002.

Eric Joanis, Suzanne Stevenson, and David James. A general feature space for automatic verb classification. *Natural Language Engineering*, 2008.

Karin Kipper-Schuler. *VerbNet: A broad-coverage, comprehensive verb lexicon*. 2005.

Anna Korhonen, Yuval Krymolowski, and Ted Briscoe. A large subcategorization lexicon for natural language processing applications. In *Proc. of the 5th LREC*, 2006.

Anna Korhonen, Yuval Krymolowski, and Nigel Collier. The Choice of Features for Classification of Verbs in Biomedical Texts. In *Proc. of COLING*, 2008.

Claudia Kunze and Lothar Lemnitzer. GermaNet-representation, visualization, application. In *Proc. of LREC*, 2002.

Lillian. Lee. On the effectiveness of the skew divergence for statistical language analysis. In *Artificial Intelligence and Statistics*, 2001.

Geoffrey Leech. 100 million words of english: the british national corpus. *Language Research*, 1992.

Beth. Levin. English verb classes and alternations: A preliminary investigation. *Chicago, IL*, 1993.

Jianguo Li and Chris Brew. Which Are the Best Features for Automatic Verb Classification. In *Proc. of ACL*, 2008.

Diana McCarthy. *Lexical Acquisition at the Syntax-Semantics Interface: Diathesis Alternations, Subcategorization Frames and Selectional Preferences*. PhD thesis, University of Sussex, UK, 2001.

Marina. Meila. The multicut lemma. Technical report, University of Washington, 2001.

Marina Meila and Jianbo Shi. A random walks view of spectral segmentation. AISTATS, 2001.

George A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 1995.

Pedro J. Moreno, Purdy P. Ho, and Nuno Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In *Proc. of NIPS*, 2004.

Andrew Y. Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Proc. of NIPS*, 2002.

Diarmuid Ó Séaghdha and Ann Copestake. Semantic classification with distributional kernels. In *Proc. of COLING*, 2008.

Judita Preiss, Ted Briscoe, and Anna Korhonen. A system for large-scale acquisition of verbal, nominal and adjectival subcategorization frames from corpora. In *Proc. of ACL*, 2007.

Jan Puzicha, Thomas Hofmann, and Joachim M. Buhmann. A theory of proximity based clustering: Structure detection by optimization. *Pattern Recognition*, 2000.

Sabine Schulte im Walde. Experiments on the automatic induction of german semantic verb classes. *Computational Linguistics*, 2006.

Sabine Schulte im Walde, Christian Hying, Christian Scheible, and Helmut Schmid. Combining EM Training and the MDL Principle for an Automatic Verb Classification incorporating Selectional Preferences. In *Proc. of ACL*, pages 496–504, 2008.

Lei Shi and Rada Mihalcea. Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing. In *Proc. of CICLING*, 2005.

Suzanne Stevenson and Eric Joanis. Semi-supervised verb class discovery using noisy features. In *Proc. of HLT-NAACL 2003*, pages 71–78, 2003.

Lin Sun, Anna Korhonen, and Yuval Krymolowski. Verb class discovery from rich syntactic data. *Lecture Notes in Computer Science*, 4919:16, 2008.

Robert Swier and Suzanne Stevenson. Unsupervised semantic role labelling. In *Proc. of EMNLP*, 2004.

Deepak Verma and Marina Meila. Comparison of spectral clustering methods. *Advances in Neural Information Processing Systems (NIPS 15)*, 2003.

Andreas Vlachos, Anna Korhonen, and Zoubin Ghahramani. Unsupervised and constrained dirichlet process mixture models for verb clustering. In *Proc. of the Workshop on Geometrical Models of Natural Language Semantics*, 2009.

Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2007.

Beñat Zapirain, Eneko Agirre, and Lluís Màrquez. Robustness and generalization of role sets: PropBank vs. VerbNet. In *Proc. of ACL*, 2008.

Improving Web Search Relevance with Semantic Features

Yumao Lu Fuchun Peng Gilad Mishne Xing Wei Benoit Dumoulin
Yahoo! Inc.
701 First Avenue
Sunnyvale, CA, 94089
yumaol,fuchun,gilad,xwei,benoitd@yahoo-inc.com

Abstract

Most existing information retrieval (IR) systems do not take much advantage of natural language processing (NLP) techniques due to the complexity and limited observed effectiveness of applying NLP to IR. In this paper, we demonstrate that substantial gains can be obtained over a strong baseline using NLP techniques, if properly handled. We propose a framework for deriving semantic text matching features from named entities identified in Web queries; we then utilize these features in a supervised machine-learned ranking approach, applying a set of emerging machine learning techniques. Our approach is especially useful for queries that contain multiple types of concepts. Comparing to a major commercial Web search engine, we observe a substantial 4% DCG5 gain over the affected queries.

1 Introduction

Most existing IR models score documents primarily based on various term statistics. In traditional models—from classic probabilistic models (Croft and Harper, 1979; Fuhr, 1992), through vector space models (Salton et al., 1975; Narita and Ogawa, 2000), to well studied statistical language models (Ponte and Croft, 2000; Lafferty and Zhai, 2001)—these term statistics have been captured directly in the ranking formula. More recently, *learning to rank* approaches to IR (Friedman, 2002) have become prominent; in these frameworks, that aim at learning a ranking function from data, term statistics are often modeled as *term matching features* in a machine learning process.

Traditional text matching features are mainly based on frequencies of n-grams of the user's query in a variety of document sections, such as the document title, body text, anchor text, and so on. Global information such as frequency of term or term group in the corpus may also be used, as well as its combination with local statistics – producing relative scores such as $tf \cdot idf$ or BM25 scores (Robertson et al., 1995). Matching may be restricted to certain window sizes to enforce proximity, or may be more lenient, allowing unordered sequences and nonconsecutive sequences for a higher recall.

Even before machine learning was applied to IR, NLP techniques such as Named Entity Recognition (NER), Part-of-Speech (POS) tagging, and parsing have been applied to both query modeling and document indexing (Smeaton and van Rijsbergen, 1988; Narita and Ogawa, 2000; Sparck-Jones, 1999). For example, statistical concept language models generalize classic n-gram models to concept n-gram model by enforcing query term proximity within each concept (Srikanth and Srihari, 2003). However, researchers have often reported limited gains or even decreased performance when applying NLP to IR (Voorhees, 1999).

Typically, concepts detected through NLP techniques either in the query or in documents are used as proximity constraints for text matching (Sparck-Jones, 1999), ignoring the actual concept type. The machine learned approach to document ranking provides us with an opportunity to revisit the manner in which NLP information is used for ranking. Using knowledge gained from NLP application as features rather than heuristically allows us much greater flexibility in the amount and variability of information used – e.g., incorporating knowledge about the actual entity types. This has several benefits: first, entity types appearing in queries are an indicator of the user's intent. A query consisting of a business *category* and a location (e.g., *hotels Palo Alto*) appears to be

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 648–657,
Singapore, 6-7 August 2009.

informational, and perhaps is best answered with a page containing a list of hotels in Palo Alto. Queries containing a business *name* and a location (e.g., *Fuki Sushi Palo Alto*) are more navigational in nature – for many users, the intent is finding the home page of a specific business. Similarly, entity types appearing in documents are an indicator of the document type. For example, if "Palo Alto" appears ten times in document's body text, it is more likely to be a local listing page than a home page. For the query *hotels Palo Alto*, a local listing page may be a good page, while for the query *Fuki Sushi Palo Alto* a listing page is not a good page.

In addition, knowledge of the particular entities in queries allows us to incorporate external knowledge about these entities, such as entity-specific stopwords ("inc." as in *Yahoo Inc.* or "services" as in *kaiser medical service*), and so on.

Finally, even when using named entities only for deriving proximity-related features, we can benefit from applying different levels of proximity for different entities. For example, for entities like cities (e.g., "River Side"), the proximity requirement is fairly strict: we should not allow extra words between the original terms, and preserve their order. For other entities the proximity constraint can be relaxed—for example, for person names, due to the middle name convention: *Hillary Clinton* vs. *Hillary R. Clinton*.

In this paper, we propose a systematic approach to modeling semantic features, incorporating concept types extracted from query analysis. Vertical attributes, such as city-state relationships, metropolitan definition, or idf scores from a domain specific corpus, are extracted for each concept type from vertical database. The vertical attributes, together with the concept attributes, are used to compose a set of semantic features for machine learning based IR models. A few machine learning techniques are discussed to further improve relevance for subclass of difficult queries such as queries containing multiple types of concepts. Figure 1 shows an overview of our approach; after discussing related work in Section 2, we spend Sections 3 to 5 of the paper describing the components of our system. We then evaluate the effectiveness of our approach both using general queries and with a set of "difficult" queries; our results show that the techniques are robust, and particularly effective for this type of queries. We conclude in Section 7.

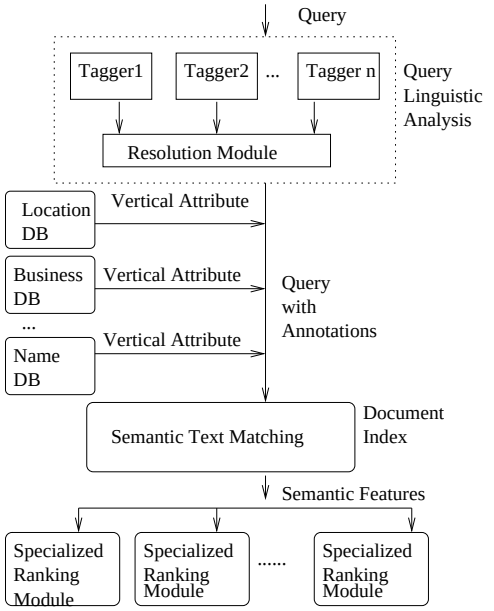


Figure 1: Ranking with Semantic Features

2 Related Work

There is substantial body of work involving usage of NLP techniques to improve information retrieval (Brants, 2003; Strzalkowski et al., 1996). Allan and Ragahavan (Allan and Raghavan, 2002) use Part-of-Speech tagging to reduce ambiguity of difficult queries by converting short queries to questions. In other POS-tags work, Arampatzis *et al.* (Arampatzis et al., 1990) observed an improvement when using nouns only for retrieval. Croft et al. (Croft et al., 1991) and Tong et al. (Buckley et al., 1993; Tong et al., 1996) explored phrases and structured queries and found phrases are effective in improving retrieval performance. Voorhees (Voohees, 1993) uses word sense disambiguation to improve retrieval performance. One IR domain that consistently benefits from usage of various NLP techniques is question answering, where queries are formed in natural language format; e.g., (Peng et al., 2005).

In general, however, researchers often observe limited gains or even degraded performance when applying NLP to IR (Voorhees, 1999). Having said this, most past studies use small datasets and a modest baseline; it is unclear whether a similar conclusion would be reached when using a state-of-art system such as a commercial web search engine as a baseline, and a full-web corpus – as we do in this paper. This leads to another difference between this work and existing work involving named entity recognition for retrieval. Most

previous research on usage of named entities in IR combines entity detection in documents and queries (Prager et al., 2000). Entity detection in document has a high indexing cost that is often overlooked, but cannot be ignored in the case of commercial search engines. For this reason, we restrict NLP processing to queries only – although we believe that document-side NLP processing will provide additional useful information.

3 Query Analysis

We begin by briefly describing our approach to named entity recognition in web queries, which serves as the basis for deriving the semantic text matching features.

Named entity recognition (NER) is the task of identifying and classifying entities, such as person names or locations, in text. The majority of state-of-the-art NER methods utilize a statistical approach, attempting to learn a mapping between a sequence of observations (words) and a sequence of tags (entity types). In these methods, the sequential nature of the data is often central to the model, as named entities tend to appear in particular context in text. For example, for most types of text, in the two sequences *met with X* and *buy the Y*, the likelihood of X being a person name is substantially higher than the corresponding likelihood of Y. Indeed, many named entity taggers perform well when applied to grammatical text with sufficient contexts, such as newswire text (Sang and Meulder, 2003).

Web queries, however, tend to be short, with most queries consisting of 1–3 words, and lack context – posing a particular challenge for identifying named entities in them. Existing work on NER in web queries focuses on tailoring a solution for a particular entity type and its usage in web search (Wang et al., 2005; Shen et al., 2008); in contrast, we aim at identifying a large range of possible entities in web queries, and using a generic solution for all of them.

In web queries, different entity types may benefit from different detection techniques. For example, an entity type with a large variability among instances as well as existence of external resources like product name calls for an approach that can make use of many features, such as a conditional random field; for entity types that are more structured like person names, a grammar-based approach can be more effective (Shen et al., 2008). To this end, we utilize multiple approaches for entity detection and combine them into a single, coherent "interpretation" of the query.

Given a query, we use several entity recognizers in parallel, one for each of the common entity types found in web queries. The modeling types may differ between the recognizers: some are Markovian models, while others are just dictionary lookups; the accuracy of each recognizer is also different. We then have a machine-learned disambiguation module that combines output from different taggers, ranking the tagging sequences. The details of scoring is out of the scope of this paper, and we omit it for simplicity.

4 Semantic Text Matching Features

Our proposed semantic features operate at the semantic type level rather than at the term level: instead of matching a term (or set of terms) in documents, we match their semantic type. Given the query *San Francisco colleges* and the annotation $[San\ Francisco]_{CityName}\ [colleges]_{BusinessCategory}$, the semantic text matching features would describe how relevant a document section is for a entity of type $CityName$, for $BusinessCategory$, and for their combination.

Concretely, we exploit a set of features that attempts to capture proximity, general relevance, and vertical relevance for each type of semantic tag and for each section of the document. We now review these feature by their broad types.

4.1 Semantic Proximity Features

Proximity features—features that capture the degree to which search terms appear close to each other in a document—are among the most important feature sets in ranking functions. Traditional proximity features are typically designed for all query terms (Metzler and Croft, 2005) and may suffer from wrong segmentations of the query. For example, for the query *New York city bus charter*, a traditional proximity feature may treat "city bus" similarly to "York city." But given detailed information about the entities in the query in their types, we can enforce proximity for "New York city" and "bus charter" more accurately. Different types of entities usually have different proximity characteristics in relevant documents. Strongly-bound entities such as city names typically have very high proximity in relevant documents, while entities such as business names may have much

lower proximity: a search for *Kaiser medical office*, for example, may be well-served with documents referring to *Kaiser Permanente medical office*, and as we mentioned before, person names matches may also benefit from lenient proximity enforcement. This is naturally addressed by treating each entity type differently.

We propose a set of semantic proximity features that associate each semantic tag type with generic proximity measures. We also consider tagging confidence together with term group proximity; we discuss these two approaches next.

4.1.1 Semantic Minimum Coverage (SMC)

Minimum Coverage (MC) is a popular span based proximity distance measure, which is defined as the length of the shortest document segment that cover the query term at least once in a document (Tao and Zhai, 2007). We extend this measure to Semantic Minimum Coverage (SMC) for each semantic type t in document section s and define it as

$$\text{SMC}_{t,s} = \frac{1}{|\{k|T_k = t\}|} \sum_{i \in \{k|T_k=t\}} w_i \text{MC}_{i,s},$$

where w_i is a weight for tagged term group i, $\text{MC}_{i,s}$ is the the minimum coverage of term group i in document section s, $\{k|T_k = t\}$ denotes the set of all concepts having type t, and $|\{k|T_k = t\}|$ is the size of the set. The definition of the weight w is flexible. We list a few candidate weighting schemes in this paper: uniform weights (w^u), weights based on idf scores (w^{idf}) and "strength"-based weight (w^s), which we define as follows:

$$w^u = 1;$$
$$w^{\text{idf}} = \frac{c}{f_q}$$

where c is a constant and f_q is the frequency of the term group in a large query log;

$$w^s = \min_l \text{MI}_l$$

where MI_l is the point-wise mutual information of the l-th consecutive pair within the semantic tag. We can also combine strength and idf scores such that the weight reflects both relative importance and constraints in proximity. In this paper, we use

$$w^{si} = w^s w^{\text{idf}}.$$

In Section 6, we use all four weighting schemes mentioned above in the semantic feature set.

4.1.2 Semantic Moving Average BM25 (SMABM25)

BM25, a commonly-used bag-of-words relevance estimation method (Robertson et al., 1995), is defined (when applied to document sections) as

$$\text{BM25} = \sum_j \text{idf}_j \frac{f_{j,s}(c_1+1)}{f_{i,s} + c_1(1 - c_2 + c_2 \frac{l_s}{\bar{l_s}})}$$

where $f_{j,s}$ is the frequency of term j in section s, l_s is the length of section s, $\bar{l_s}$ is the average length of document section s, c_1, c_2, c_3 are constants and the idf score of term j is defined as

$$\text{idf}_j = \log \frac{c_4 - d_j + c_5}{d_j + c_5},$$

where d_j is the number of sections in all collections that contains term j and c_4, c_5 are constants.

To characterize proximity, we could use a fixed length sliding window and calculate the average BM25. We further associate each sliding average BM25 with each type of semantic term groups. This results in a Semantic Moving Average BM25 (SMABM25) of type t, which we define as follows:

$$\frac{1}{|\{k|T_k = t\}|} \sum_{i \in \{k|T_k=t\}} (1/M) \sum_m BM25_m$$

where m is a fixed length sliding window m and M is the total number of sliding windows (that depends on the length of the section window size).

4.2 Semantic Vertical Relevance Features

Vertical databases contain a large amount of structured domain knowledge typically discarded by traditional web relevance features. Having access to the semantic types in queries, we can tap into that knowledge to improve accuracy. For example, term frequencies in different corpora can assist in determining relevance given an entity type. As we mentioned in Section 1, we observe that term frequency in a database of business names provides an indication of the business brand, the key part of the business name phrase. While both "yahoo" and "inc" are very common terms on the web, in a database of businesses only "inc" is common enough to be considered a stopword in the context of business names.

We propose a Vertical Moving Average BM25 (VMABM25) as a feature aiming at quantifying the vertical knowledge for web search. The basic idea here is to replace the idf score idf_j of

SMABM25 with an idf score calculated from a vertical database for type t, namely idf_j^t:

$$\frac{1}{|\{k|T_k = t\}|} \sum_{i \in \{k|T_k=t\}} (1/M) \sum_m \text{BM25}_{m,t}$$

where

$$\text{BM25}_{m,t} = \sum_j \text{idf}_j^t \frac{f_{j,s}(c1+1)}{f_{i,s} + c_1(1 - c_2 + c_2 \frac{l_s}{l_s})}$$

where the idf_j^t is associated with the semantic type t and calculated from the corpus associated with that type.

VMABM25 links vertical knowledge, proximity, and page relevance together; we show later that it is one of most salient features among all semantic features.

4.3 Generalized Semantic Features

Finally, we develop a generalized feature based on the previous features by removing tags. Semantic features are often sparse, as many queries contain one entity or no entities at all; generalized features increase their coverage by combining the basic semantic features. An entity without tag is essentially a segment.

A segment feature x_i for query i does not have entity type and can be expressed as

$$x_i = \frac{1}{K_i} \sum_{k=1}^{K_i} x_{T(k)}$$

where K_i is the number of segments in the query and $T(k)$ is the semantic type associated with kth concept.

Although these features are less informative than type-specific features, one advantage of using them is that they have substantially higher coverage. In our experiments, more than 40% of the queries have some identified entity. Another relatively subtle advantage is that segment features have no type related errors: the only possible error is a mistake in entity boundaries.

5 Ranking Function Optimization

The ultimate goal of the machine learning approach to web search is to learn a ranking function $h(x_i)$, where x_i is a feature vector of a query-document pair i, such that the error

$$L(h) \equiv \sum_{i=1}^{N} (y_i - h(x_i))^2 \qquad (1)$$

is minimized. Here, y_i is the actual relevance score for the query-document pair i (typically assigned by a human) and N is the number of training samples.

As mentioned in the previous Section, an inherent issue with semantic features is their sparseness. User queries are usually short, with an average length of less than 3 words. Text matching features that are associated with the semantic type of query term or term groups are clearly sparse comparing with traditional, non-entity text matching features – that can be derived for any query. When a feature is very sparse, it is unlikely that it would play a very meaningful role in a machine learned ranking function, since the error L would largely depend on other samples that do not contain the specific semantic features at all. To overcome the spareness issue and take advantage of semantic features, we suggested generalizing our features; but we also exploit a few ranking function modeling techniques.

First, we use a "divide-and-conquer" approach. Long queries usually contain multiple concepts and could be difficult to retrieve relevant documents. Semantic features, however, are rich in this set of queries. We may train special models to further optimize our ranking function for those queries. The loss function over ranking function h becomes

$$L_C(h) \equiv \sum_{i \in C} (y_i - h(x_i))^2 \qquad (2)$$

where C is the training set that falls into a predefined subclass. For example, queries containing both location and business name, queries contains both location and business category, etc, are good candidates to apply semantic features.

To this end, we first classify queries into several classes, each of which has multiple types of entities. The semantic features of those types would be dense for this subclass of queries. We then train models that may rank the specific class of queries well. This approach, however, may suffer from significantly less training samples due to training data partition resulted from the query classification. Increasing the modeling accuracy, then, comes at a cost of reduced data available for training. We apply two techniques to address this issue. The first approach is to over-weight subclass training samples such that the subclass of queries plays a more important role in modeling while still

keeping a large pool of the overall training samples. The second approach is model adaptation: a generalized incremental learning method. Here, instead of being over-weighted in a joint optimization, the subclass of training data is used to modify an existing model such that the new model is "adapted" to the subclass problem. We elaborate on our approaches as follows.

5.1 Weighted Training Samples

To take advantage of both large a pool of training samples and sparse related semantic features for a subclass of queries, we could modify the loss function as follows

$$L_C^w(h) \equiv w \sum_{i \in C} (y_i - h(x_i))^2 + \sum_{i \in \bar{C}} (y_i - h(x_i))^2, \quad (3)$$

where $\bar{C}$ is the complement of set C. Here, the weight w is a compromise between loss function (1) and (2). When $w = 1$, we have

$$L_C^1(h) \equiv L(h);$$

when $w -> \infty$

$$L_C^\infty(h) \equiv L_C(h).$$

A large weight may help optimize the training for a special subclass of queries, and a small weight may help to preserve good generality of the ranker. We could use cross-validation to select the weight w to optimize a the ranking function for a subclass of queries. In practice, a small w is desired to avoid overfitting.

5.2 Model Adaptation

Model adaptation is an emerging machine learning technique that is used for information retrieval applications with limited amount of training data. In this paper, we apply Trada, proposed by Chen et al. (Chen et al., 2008), as our adaptation algorithm.

The Trada algorithm aims at adapting tree-based models. A popular tree based regression approach is Gradient Boosting Trees (GBT) , which is an additive model $h(x) = \sum_{k=1}^{K} \gamma_k h_k(x)$, where each regression tree h_k is sequentially optimized with a hill-climbing procedure. As with other decision trees, a binary regression tree $h_k(x)$ consists of a set of decision nodes; each node is associated with a feature variable and a splitting value that partition the data into two parts, with the corresponding predicted value defined in the leave node. The basic idea of Trada is to apply piecewise linear transformation to the base model based on the new training data. A set of linear transformations are applied to each decision node, either predict or split point or both, such that the new predict or the split point of a node in a decision tree satisfies

$$v = (1 - p_C)\hat{v} + p_C v_C$$

where $\hat{v}$ denotes predict or split point of that node in the base mode and v_C denotes predict or split point of that node using new data set C, and the weight p_C depends on the number of original training data and new training data that fall through the node. For each node, the split or predict can be estimated by

$$p_C = \frac{\beta n_C}{n + \beta n_C},$$

where n is the number of training sample of the base model that fall through the node, n_C is the number of new training sample that fall through the node, and β is a parameter that can be determined using cross validation. The parameter β is used to over-weight new training data, an approach that is very effective in practice. For new features that are not included in the base model, more trees are allowed to be added to incorporate them.

6 Experiments

We now measure the effectiveness of our proposal, and answer related questions, through extensive experimental evaluation. We begin by examining the effectiveness of features as well as the modeling approaches introduced in Section 5 on a particular class of queries—those with a local intent. We proceed by evaluating whether if the *type* associated with each entity really matters by comparing results with type dependent semantic features and segment features. Finally, we examine the robustness of our features by measuring the change in the accuracy of our resulting ranking function when the query analysis is wrong; we do this by introducing simulated noise into the query analysis results.

6.1 Dataset

Our training, validation and test sets are human-labeled query-document pairs. Each item in the

sets consists of a feature vector $\mathbf{x_i}$ representing the query and the document, and a judgment score y_i assigned by a human. There are around 600 features in each vector, including both the newly introduced semantic features and existing features; features are either query-dependent ones, document-dependent ones, or query-and-document-dependent features.

The training set is based on uniformly sampled Web queries from our query log, and top ranked documents returned by commercial search engines for these queries; this set consists of 1.24M query-document pairs.

We use two additional sets for validation and testing. One set is based on uniformly sampled Web queries, and contains 42790 validation samples and 70320 test samples. The second set is based on uniformly sampled *local queries*. By local queries, we mean queries that contain at least two types of semantic tags: a location tag (such as street, city or state name) and a business tag (a business name or business category). We refer to this class of queries "local queries," as users often type this kind of queries in local vertical search. The local query set consists of 11040 validation samples and 39169 test samples. In the training set we described above, there are 56299 training samples out of the 1.24M total number of training samples that satisfy the definition of local queries. We call this set local training subset.

6.2 Evaluation Metrics

To evaluate the effectiveness of our semantic features we use Discounted Cumulative Gain (DCG) (Jarvelin and Kekalainen, 2000), a widely-used metric for measuring Web search relevance. (Jarvelin and Kekalainen, 2000). Given a query and a ranked list of K documents ($K = 5$ in our experiments), the DCG for this query is defined as

$$\mathrm{DCG}(K) = \sum_{i=1}^{K} \frac{y_i}{\log_2(1+i)}. \qquad (4)$$

where $y_i \in [0, 10]$ is a relevance score for the document at position i, typically assigned by a human, where 10 is assigned to the most relevant documents and 0 to the least relevant ones.

To measure statistical significance, we use the Wilcoxon test (Wilcoxon, 1945); when the p-value is below 0.01 we consider a difference to be statistically significant and mark it with a **bold font** in the result table.

6.3 Experimental Results

We use Stochastic Gradient Boosting Trees (SGBT) (Friedman, 2002), a robust none linear regression algorithm, for training ranking functions and, as mentioned earlier, Trada (Chen et al., 2008) for model adaptation.

Training parameters are selected to optimize the relevance on a separated validation set. The best resulting is evaluated against the test set; all results presented here use the test set for evaluation.

6.3.1 Feature Effectiveness with Ranking Function Modeling

We apply the modeling approaches introduced in Section 5 to improve feature effectiveness on "difficult" queries—those more than one entity type; we evaluate these approaches with the semantic-feature-rich set, the local query test set. We split training sets into two parts: one set belongs to the local queries, the other is the rest. We first weight the local queries and use the combined dataset as training data to learn the ranking functions; we train functions with and without the semantic features. We evaluate these functions against the local query test set. The results are summarized in Table 1, where w denotes the weight assigned to the local training set, bolded numbers are statistically significant result compared to the baseline, uniformly weighted training data without semantic features (with superscript b). It is interesting to observe that without semantic features, over-weighted local training data does not have statistically significant impact on the test performance; with semantic features, a proper weight over training samples does improve test performance substantially.

Table 1: Evaluation of Ranking Models Trained Against Over-weighted Local Queries with Semantic Features on the Local Query Test Set

	w/o semantic features		w/ semantic features	
Weight	DCG(5)	Impr.	DCG(5)	Impr.
$w = 0$	8.09^b	-	8.25	$\mathbf{2.0\%}$
$w = 2$	8.09	0.02%	8.26	$\mathbf{2.1\%}$
$w = 4$	8.13	0.49%	8.34	$\mathbf{3.1\%}$
$w = 8$	8.13	0.49%	8.42	$\mathbf{4.1\%}$
$w = 16$	8.13	0.49%	8.30	$\mathbf{2.6\%}$
$w = 32$	8.04	-0.60%	8.27	$\mathbf{2.2\%}$

Next, we use the local query training set as "new data" in the tree adaptation approach. In tree adaptations, all parameters are set to optimize the performance over the local validation set. We com-

pare two major adaptation approaches proposed in (Chen et al., 2008): adapting predict only and adapting both predict and split. We use the model trained with the combined training and uniform weights as the baseline; results are summarized in Table 2.

Table 2: Trada Algorithms with Semantic Features on Local Query Test Set

	w/o semantic feat.		w/ semantic feat.	
Ada. Appr.	DCG(5)	Impr.	DCG(5)	Impr.
Combined data	8.09^b	-	8.25	$\mathbf{2.0}\%$
Ada. predict	8.02	-0.1%	8.14	0.6%
Ada. predict & split	8.00	-0.1%	8.17	$\mathbf{1.0}\%$

Comparing Tables 1 and 2, note that using the combined training data with local query training samples over-weighted achieves better results than tree adaption. The latter approach, however, has the advantage of far less training time, since the adaptation is over a much smaller local query training set. With the same hardware, it takes just a few minutes to train an adaptation model, while it takes days to train a model over the entire, combined training data. Considering that massive model validation tasks are required to select good training parameters, training many different models with over a million training samples becomes prohibitly costly. Applying tree adaptation techniques makes research and prototyping of these models feasible.

6.3.2 Type Dependent Semantic Features vs. Segment Features

Our next experiment compares type-dependent features and segment features, evaluating models trained with these features against the local query test set. No special modeling approach is applied here; results are summarized in Table 3. We observe that by using type-dependent semantic features only, we can achieve as much as by using all semantic features. Since segment features only convey proximity information while the base feature set already contain a systematic set of proximity measures, the improvement through segment features is not as significant as the the type dependent ones.

6.3.3 Robustness of Semantic Features

Our final set of experiments aims at evaluating the robustness of our semantic features by introducing

Table 3: Type-dependent Semantic Features vs. Segment Features

Feature set	DCG(5)
base + type dependent semantic features	8.23
base + segment features	8.19
base + all semantic features	8.25

simulated errors to the output of our query analysis. Concretely, we manipulate the precision and the recall of a specific type of entity tagger, t, on the training and test set. To decrease the *recall* of type t, we uniformly remove a set of $a\%$ tags of type t – preserving precision. To decrease *precision*, we uniformly select a set of query segments (viewing the entity detection as simple segmentation, as detailed earlier) and assign the semantic type t to those segments. Since the newly added term group are selected from query segmentation results, the introduced errors are rather semantic *type* error than boundary error or proximity error. The total number of newly assigned type t tags are $b\%$ of the original number of type t tags in the training set. By doing this, we decrease the precision of type t while keeping the recall of it at the same level.

Suppose the original tagger achieves precision p and recall r. By removing $a\%$ of tags, we have estimated precision $\hat{p}$ and recall $\hat{r}$ defined as follows:

$$\hat{r} = \frac{100r - ar}{100},$$

$$\hat{p} = p.$$

By adding $b\%$ more term group to this type, we have estimated precision and recall as

$$\hat{p} = \frac{100p}{100 + bp},$$

$$\hat{r} = r.$$

In the experiment reported here we use BUSINESS NAME as the target semantic type for this robustness experiment. An editorial test shows that our tagger achieves 74% precision and 66% recall based on a random set of human labeled queries for this entity type. We train ranking models with various values of a and b. When we reduce the estimated recall, we evaluate these models against the local test set since other data are not affected. The results are summarized in Table 4.

When we reduce the precision, we evaluate the resulting models against the general test set as

Table 4: Relevance with simulated error on local query test set

a	b	$\hat{p}$	$\hat{r}$	DCG(5)	Impr.
0	0	0.74	0.66	8.25	–
10	0	0.74	0.594	8.21	0.48%
20	0	0.74	0.528	8.19	**0.72%**
40	0	0.74	0.396	8.18	**0.85%**

Table 5: Search relevance with simulated error for semantic features on general test set

a	b	$\hat{p}$	$\hat{r}$	DCG(5)	Impr.
0	0	0.74	0.66	10.11	-
0	10	0.689	0.66	10.11	0.00%
0	20	0.645	0.66	10.12	0.10%
0	40	0.571	0.66	10.12	0.10%
0	60	0.513	0.66	10.12	0.10%
0	80	0.465	0.66	10.11	0.00%
0	100	0.425	0.66	10.10	−0.10%

simulated errors would virtually affect any samples with certain probability. Results appear in Table 5. The results are quite interesting: when the recall of business name entity decreases, we observe statistically significant relevance degradation: if less entities are discovered, search relevance is hit. The experiments with simulated precision error, however, are less conclusive. One may note the experiments are conducted over the general test set. Therefore, it is not clear if the precision of the NER system really has insignificant impact on the IR relevance or just the impact is diluted in a larger test set.

6.4 Case Analysis

In this section, we take a close look at a few cases where our new semantic features help most and where they fail. For the query *silverado ranch in irving texas*, with no semantic features, the ranking function ranks a local listing page for this business, `http://local.yahoo.com/info-28646193`, as the top document. With semantic features, the ranking function ranks the business home page: `http://www.silveradoranchparties.com/` as top URL. Examining the two documents, the local listing page actually contains much more relevant anchor text, which are the among the most salient features in traditional ranking models. The home page, however, contains almost no relevant anchor text: for a small business home page, this is not a rare situation. Looking at the semantic features of these two pages, the highest resolution of location, the city name "Irving," appears in the document body text 19 times in the local listing page body text, and only 2 times in the home page body text. The training process learns, then, that for a query for a local business name (rather than a business category), home pages—even with fewer location terms in them—are likely to be more relevant than a local listing page that usually contain high frequency location terms.

In some cases, however, our new features do hurt performance. For the query *pa treasurers office*, the ranking function with no semantic features ranks the document `http://www.patreasury.org` highest, while the one with semantic features ranks the page `http://www.pikepa.org/treasurer.htm` higher. The latter page is somewhat relevant: it is a treasurer's office in Pennsylvania. However, it belongs to a specific county, which makes it less relevant than the former page. This is a classic error that we observe: a mismatch of the intended location area. While users are looking for state level business, we provide results of county level. To resolve this type of error, query analysis and semantic text matching are no longer enough: here, the ranking function needs to know that Pike County is a county in Pennsylvania, Milford is a city in Pike County, and neither are referred to by the user. Document-side entity recognition, however, may provide this type of information, helping to address this type of errors.

7 Conclusion and Future Research

In this paper, we investigate how semantic features can improve search relevance in a large-scale information retrieval setting; to our knowledge, it is the first study of this approach on a web scale. We present a set of features that incorporate semantic and vertical knowledge into the retrieval process, propose techniques to handle the sparseness problem for these features, and describe how they fit in the learning process. We demonstrate that these carefully designed features significantly improve relevance, particularly for difficult queries – long queries with multiple entities.

The work reported here focuses on query-side processing, avoiding the indexing cost of document processing. We are currently investigating document-side analysis to complement the query-side work, and believe that this will further boost the retrieval accuracy; we hope to report on this in a follow-up study.

References

J. Allan and H. Raghavan. 2002. Using Part-of-Speech Patterns to Reduce Query Ambiguity. In *Proceedings of SIGIR*.

A. T. Arampatzis, Th. P. Weide, C. H. A. Koster, and P. Bommel. 1990. Text Filtering using Linguistically-motivated Indexing Terms. Technical Report CSI-R9901, Computing Science Institute, University of Nijmegen, Nijmegen,The Netherlands.

Thorsten Brants. 2003. Natural Language Processing in Information Retrieval. In *Proceedings of CLIN*.

C. Buckley, J. Allan, and G. Salton. 1993. Automatic Routing and Ad-hoc Retrieval Using SMART: TREC 2. In *Proceedings of TREC-2*.

Keke Chen, Rongqing Lu, C.K. Wong, Gordon Sun, Larry Heck, and Belle Tseng. 2008. Trada: tree based ranking function adaptation. In *Proceedings of CIKM*.

W.B. Croft and D.J. Harper. 1979. Using Probabilistic Models of Document Retrieval without Relevance Information. *Journal of Documentation*, 37:285–295.

W. Bruce Croft, Howard R. Turtle, and David D. Lewis. 1991. The Use of Phrases and Structured queries in Information Retrieval. In *Proceedings of SIGIR*.

J. H. Friedman. 2002. Stochastic Gradient Boosting. *Computational Statistics and Data Analysis*, 38(4):367–378.

N. Fuhr. 1992. Probabilistic Models in Information Retrieval. *The Computer Journal*, 35:243–255.

K. Jarvelin and J. Kekalainen. 2000. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *Proceedings of SIGIR*.

J. Lafferty and C. Zhai. 2001. Document Language Models, Query Models and Risk Minimization for Information Retrieval. In *Proceedings of SIGIR*.

Donald Metzler and W. Bruce Croft. 2005. A markov random field model for term dependencies. In *Proceedings of SIGIR*.

M. Narita and Y. Ogawa. 2000. The Use of Phrases from Query Texts in Information Retrieval. In *Proceedings of SIGIR*.

Fuchun Peng, Ralph Weischedel, Ana Licuanan, and Jinxi Xu. 2005. Combining Deep Linguistics Analysis and Surface Pattern Learning: A Hybrid Approach to Chinese Definitional Question Answering. In *In Proceedings of the HLT-EMNLP*.

J. Ponte and W.B. Croft. 2000. A Language Modeling Approach to Informaiton Retrieval. In *Proceedings of SIGIR*.

John Prager, Eric Brown, Anni Coden, and Dragomir Radev. 2000. Question-answering by Predictive Annotation. In *Proceedings of SIGIR*.

Stephen Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1995. Okapi at TREC-3. In *Proceedings of TREC-3*.

G. Salton, C. S. Yang, and C. T. Yu. 1975. A Theory of Term Importance in Automatic Text Analysis. *Journal of the Ameican Society of Information Science*, 26:33–44.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL*.

Dou Shen, Toby Walkery, Zijian Zheng, Qiang Yang, and Ying Li. 2008. Personal Name Classification in Web Queries. In *Proceedings of WSDM*.

A.F. Smeaton and C.J. van Rijsbergen. 1988. Experiments on Incorporating Syntactic Processing of User Queries into a Document Retrieval Stragegy. In *Proceedings of SIGIR*.

K. Sparck-Jones, 1999. *What is the Role of NLP in Text Retrieval*, pages 1–25. Kluwer.

M. Srikanth and R.K. Srihari. 2003. Incorporating Query Term Dependencies in Language Models for Document Retrieval. In *Proceedings of SIGIR*.

Tomek Strzalkowski, Jose Perez-Carballo, Jussi Karlgren, Anette Hulth, Pasi Tapanainen, and Timo Lahtinen. 1996. Natural Language Information Retrieval: TREC-8 Report. In *Proceedings of TREC-8*.

Tao Tao and ChengXiang Zhai. 2007. An exploration of proximity measures in information retrieval. In *Proceedings of SIGIR*.

X. Tong, C. Zhai, N. Millic-Frayling, and D Evans. 1996. Evaluation of Syntactic Phrase Indexing – CLARIT NLP Track Report. In *Proceedings of TREC-5*.

E. Voohees. 1993. Using WordNet to Disambiguate Word Senses for Text Retrieval. In *Proceedings of SIGIR*.

Ellen Voorhees. 1999. Natural Language Processing and Information Retrieval. *Lecture Notes in Computer Science*, 1714:32–48.

Lee Wang, Chuang Wang, Xing Xie, Josh Forman, Yansheng Lu, Wei-Ying Ma, and Ying Li. 2005. Detecting dominant locations from search queries. In *Proceedings of SIGIR*.

F. Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics*, 1:80–83.

Can Chinese Phonemes Improve Machine Transliteration?: A Comparative Study of English-to-Chinese Transliteration Models

Jong-Hoon Oh, Kiyotaka Uchimoto, and Kentaro Torisawa
Language Infrastructure Group, MASTAR Project,
National Institute of Information and Communications Technology (NICT)
3-5 Hikaridai Seika-cho, Soraku-gun, Kyoto 619-0289 Japan
{rovellia,uchimoto,torisawa}@nict.go.jp

Abstract

Inspired by the success of English grapheme-to-phoneme research in speech synthesis, many researchers have proposed phoneme-based English-to-Chinese transliteration models. However, such approaches have severely suffered from the errors in Chinese phoneme-to-grapheme conversion. To address this issue, we propose a new English-to-Chinese transliteration model and make systematic comparisons with the conventional models. Our proposed model relies on the joint use of Chinese phonemes and their corresponding English graphemes and phonemes. Experiments showed that Chinese phonemes in our proposed model can contribute to the performance improvement in English-to-Chinese transliteration.

1 Introduction

1.1 Motivation

Transliteration, i.e., *phonetic translation*, is commonly used to translate proper names and technical terms across languages. A variety of English-to-Chinese machine transliteration models has been proposed in the last decade (Meng et al., 2001; Gao et al., 2004; Jiang et al., 2007; Lee and Chang, 2003; Li et al., 2004; Li et al., 2007; Wan and Verspoor, 1998; Virga and Khudanpur, 2003). They can be categorized into those based on Chinese phonemes (Meng et al., 2001; Gao et al., 2004; Jiang et al., 2007; Lee and Chang, 2003; Wan and Verspoor, 1998; Virga and Khudanpur, 2003) and those that don't rely on Chinese phonemes (Li et al., 2004; Li et al., 2007).

Inspired by the success of English grapheme-to-phoneme research in speech synthesis, many researchers have proposed phoneme-based English-to-Chinese transliteration models. In these approaches, Chinese phonemes are generated from English graphemes or phonemes, and then the Chinese phonemes are converted into Chinese graphemes (or characters), where Chinese Pinyin strings[1] are used for representing a syllable-level Chinese phoneme sequence. Despite its high accuracy in generating Chinese phonemes from English, this approach has severely suffered from errors in Chinese phoneme-to-grapheme conversion, mainly caused by Chinese homophone confusion – one Chinese Pinyin string can correspond to several Chinese characters (Li et al., 2004). For example, the Pinyin string "LI" corresponds to such different Chinese characters as 利, 莉, and 里. For this reason, it has been reported that English-to-Chinese transliteration without Chinese phonemes outperforms that with Chinese phonemes (Li et al., 2004).

Then "Can Chinese phonemes improve English-to-Chinese transliteration, if we can reduce the errors in Chinese phoneme-to-grapheme conversion?" Our research starts from this question.

1.2 Our Approach

Previous approaches using Chinese phonemes have relied only on Chinese phonemes in Chinese phoneme-to-grapheme conversion. However, the simple use of Chinese phonemes doesn't always provide a good clue to reduce the ambiguity in Chinese phoneme-to-grapheme conversion. Let us explain with an example, the Chinese transliteration of *Greeley* in Table 1, where Chinese phonemes are represented in terms of Chinese Pinyin strings and English phonemes are represented by *ARPAbet* symbols[2].

In Table 1, Chinese Pinyin string "LI" corresponds to two different Chinese characters, 里 and

[1] Pinyin, the most commonly used Romanization system for Chinese characters, faithfully represents Chinese

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 658–667,
Singapore, 6-7 August 2009.

Table 1: Chinese Pinyin string "LI" and its corresponding Chinese characters in Chinese transliteration of *Greeley*

English grapheme	*g*	*ree*	*ley*
English phoneme	G	R IY	L IY
Chinese Pinyin	GE	LI	LI
Chinese character	格	里	利

利. It seems difficult to find evidence for selecting the correct Chinese character corresponding to each Chinese Pinyin string "LI" by just looking at the sequence of Chinese Pinyin strings "GE LI LI." However, English graphemes (*ree* and *ley*) or phonemes ("R IY" and "L IY") corresponding to Chinese Pinyin string "LI", especially their consonant parts (*r* and *l* in the English graphemes and "R" and "L" in the English phonemes), provide strong evidence to resolve the ambiguity. Thus, we can easily find rules for the conversion from Chinese Pinyin string "LI" to 里 and 利 as follows:

- ⟨ "R IY", LI ⟩ → 里
- ⟨ "L IY", LI ⟩ → 利

Based on the observation, we propose an English-to-Chinese transliteration model based on the joint use of Chinese phonemes and their corresponding English graphemes and phonemes. We define a set of English-to-Chinese transliteration models and categorize them into the following three classes:

- $\mathbf{M_I}$: **M**odels **I**ndependent of Chinese phonemes
- $\mathbf{M_S}$: **M**odels based on **S**imple use of Chinese phonemes
- $\mathbf{M_J}$: **M**odels based on **J**oint use of Chinese phonemes and English graphemes and phonemes that correspond to our proposed model.

Our comparison among the three types of transliteration models can be summarized as follows.

- The $\mathbf{M_I}$ models relying on either English graphemes or phonemes could not outperform those based on both English graphemes and phonemes.
- The $\mathbf{M_S}$ models always showed the worst performance due to the severe error rate in Chinese phoneme-to-grapheme conversion.
- The $\mathbf{M_J}$ models significantly reduced errors in Chinese phoneme-to-grapheme conversion; thus they achieved the best performance.

The rest of this paper is organized as follows. Section 2 introduces the notations used throughout this paper. Section 3 describes the transliteration models we compared. Section 4 describes our tests and results. Section 5 concludes the paper with a summary.

2 Preliminaries

Let E_G be an English word composed of *n* English graphemes, and let E_P be a sequence of English phonemes that represents the pronunciation of E_G. Let C_G be a sequence of Chinese graphemes corresponding to the Chinese transliteration of E_G, and let C_P be a sequence of Chinese phonemes that represents the pronunciation of C_G.

C_P corresponds to a sequence of the Chinese Pinyin strings of C_G. Because a Chinese Pinyin string represents the pronunciation of a syllable consisting of consonants and vowels, we divide a Chinese Pinyin string into consonant and vowel parts like "L+I", "L+I+N", and "SH+A." In this paper, we define a *Chinese phoneme* as the vowel and consonant parts in a Chinese Pinyin string (e.g., "L", "SH", and "I"). A Chinese character usually corresponds to multiple English graphemes, English phonemes, and Chinese phonemes (i.e., 里 corresponds to English graphemes *ree*, English phonemes "R IY", and Chinese phonemes "L I" in Table 1). To represent these many-to-one correspondences, we use the well-known BIO labeling scheme to represent a Chinese character, where B and I represent the beginning and inside/end of the Chinese characters, respectively, and O is not used. Each Chinese phoneme corresponds to a Chinese character with B and I labels. For example, Chinese character "里" in Table 1 can be represented as "里:B" and "里:I", where "里:B" and "里:I" correspond to Chinese phonemes "L" and "I", respectively. In this paper, we define a *Chinese grapheme* as a Chinese character represented with a BIO label, e.g., "里:B" and "里:I."

phonemes and syllables (Yin and Felley, 1990).

[2] `http://www.cs.cmu.edu/~laura/pages/arpabet.ps`

Table 2: eg_i and its corresponding ep_i, cp_i, and cg_i in *Greeley* and its corresponding Chinese transliteration "格里利"

i	1	2	3	4	5	6	7
E_G	*g*	*r*	*e*	*e*	*l*	*e*	*y*
E_P	G	R	IY	ϕ	L	IY	ϕ
C_P	GE	L	I	ϕ	L	I	ϕ
	GE	LI		ϕ	LI		ϕ
C_G	格:B	里:B	里:I	ϕ	利:B	利:I	ϕ
	格	里		ϕ	利		ϕ

Then E_P, C_P, and C_G can be segmented into a series of sub-strings, each of which corresponds to an English grapheme in E_G. We can thus write

- $E_G = eg_1, \cdots, eg_n = eg_1^n$
- $E_P = ep_1, \cdots, ep_n = ep_1^n$
- $C_P = cp_1, \cdots, cp_n = cp_1^n$
- $C_G = cg_1, \cdots, cg_n = cg_1^n$

where eg_i, ep_i, cp_i, and cg_i represent the i^{th} English grapheme, English phonemes, Chinese phonemes, and Chinese graphemes corresponding to eg_i, respectively.

Based on the definition, we model English-to-Chinese transliteration so that each English grapheme is tagged with its corresponding English phonemes, Chinese phonemes, and Chinese graphemes. Table 2 illustrates eg_i, ep_i, cp_i, and cg_i with the same example listed in Table 1 (English word *Greeley* and its corresponding Chinese transliteration "格里利")[3], where ϕ represents an empty string.

3 Transliteration Model

We defined eighteen transliteration models to be compared. These transliteration models are classified into three classes, $\mathbf{M_I}$, $\mathbf{M_S}$, and $\mathbf{M_J}$ as described in Section 1.2; each class has three basic transliteration models and three hybrid ones. In this section, we first describe the basic transliteration models in each class by focusing on the main difference among the three classes and then describe the hybrid transliteration models.

[3]We performed alignment between E_G and E_P and between E_P and C_P in a similar manner presented in Li et al. (2004). Then the two alignment results were merged using E_P as a pivot. Finally, we made a correspondence relation among eg_i, ep_i, cp_i, and cg_i using the merged alignment result and the Pinyin table.

3.1 Basic Transliteration Models

The basic transliteration models in each class are denoted as $M(x, y)$.

- $(x, y) \in X \times Y$
- $x \in X = \{E_G, E_P, E_{GP}\}$
- $y \in Y = \{\phi, C_P, JC_P\}$

x is an English-side parameter representing English grapheme (E_G), English phoneme (E_P), and the joint use of English grapheme and phoneme ($E_{GP} = \langle E_G, E_P \rangle$) that contributes to generating Chinese phonemes or Chinese graphemes in a transliteration model. y is a Chinese-phoneme parameter that represents a way of using Chinese phonemes to generate Chinese graphemes in a transliteration model. Since $M(x, \phi)$ represents a transliteration model that does not rely on Chinese phonemes, it falls into $\mathbf{M_I}$, while $M(x, C_P)$ corresponds to a transliteration model in $\mathbf{M_S}$ that only uses Chinese phonemes in Chinese phoneme-to-grapheme conversion. $M(x, JC_P)$ is a transliteration model in the $\mathbf{M_J}$ class that generates Chinese transliterations based on joint use of x and Chinese phoneme C_P, where $x \in X$. Thus, $M(x, JC_P)$ can be rewritten as $M(x, \langle x, C_P \rangle)$, where the joint representation of x and C_P, $\langle x, C_P \rangle$, is used in Chinese phoneme-to-grapheme conversion. The three basic models in $\mathbf{M_J}$ can be interpreted as follows:

- $M(E_G, JC_P) = M(E_G, \langle E_G, C_P \rangle)$
- $M(E_P, JC_P) = M(E_P, \langle E_P, C_P \rangle)$
- $M(E_{GP}, JC_P) = M(E_{GP}, \langle E_{GP}, C_P \rangle)$

$M(E_G, JC_P)$ directly converts English graphemes into Chinese phonemes without the help of English phonemes and then generates Chinese transliterations based on the joint representation of English graphemes and Chinese phonemes. The main difference between $M(E_P, JC_P)$ and $M(E_{GP}, JC_P)$ lies in the use of English graphemes to generate Chinese phonemes and graphemes. English graphemes are only used in English grapheme-to-phoneme conversion, and English phonemes play a crucial role for generating Chinese transliteration in $M(E_P, JC_P)$. Chinese phoneme-to-grapheme conversion that relies on the joint use of English graphemes, English phonemes, and Chinese

$$P_{M(E_G,JC_P)}(C_G|E_G) = \sum_{\forall C_P} P(C_P|E_G) \times P(C_G|E_G, C_P) \quad (1)$$

$$P_{M(E_P,JC_P)}(C_G|E_G) = \sum_{\forall C_P} \sum_{\forall E_P} P(E_P|E_G) \times P(C_P|E_P) \times P(C_G|E_P, C_P) \quad (2)$$

$$P_{M(E_{GP},JC_P)}(C_G|E_G) = \sum_{\forall C_P} \sum_{\forall E_P} P(E_P|E_G) \times P(C_P|E_G, E_P) \times P(C_G|E_G, E_P, C_P) \quad (3)$$

$$P_{M(E_G,C_P)}(C_G|E_G) = \sum_{\forall C_P} P(C_P|E_G) \times P(C_G|C_P) \quad (4)$$

$$P_{M(E_P,C_P)}(C_G|E_G) = \sum_{\forall C_P} \sum_{\forall E_P} P(E_P|E_G) \times P(C_P|E_P) \times P(C_G|C_P) \quad (5)$$

$$P_{M(E_{GP},C_P)}(C_G|E_G) = \sum_{\forall C_P} \sum_{\forall E_P} P(E_P|E_G) \times P(C_P|E_G, E_P) \times P(C_G|C_P) \quad (6)$$

phonemes is the key feature of $M(E_{GP}, JC_P)$. Because $M(x, JC_P)$ can be interpreted as $M(x, \langle x, C_P\rangle)$, English-side parameter x determines the English graphemes and phonemes, or both jointly used with Chinese phonemes in Chinese phoneme-to-grapheme conversion. Then we can represent the three basic transliteration models as in Eqs. (1)–(3), where $P(C_G|E_G, C_P)$, $P(C_G|E_P, C_P)$, and $P(C_G|E_G, E_P, C_P)$ are the key points in our proposed models, $\mathbf{M_J}$.

The three basic transliteration models in $\mathbf{M_S}$ – $M(E_G, C_P)$, $M(E_P, C_P)$, and $M(E_{GP}, C_P)$ – are formulated as Eqs. (4)–(6). Chinese phoneme-based transliteration models in the literature fall into either $M(E_G, C_P)$ or $M(E_P, C_P)$ (Meng et al., 2001; Gao et al., 2004; Jiang et al., 2007; Lee and Chang, 2003; Wan and Verspoor, 1998; Virga and Khudanpur, 2003). The three basic transliteration models in $\mathbf{M_S}$ are identical as those in $\mathbf{M_J}$, except for the Chinese phoneme-to-grapheme conversion method. They only depend on Chinese phonemes in Chinese phoneme-to-grapheme conversion represented as $P(C_G|C_P)$ in Eqs. (4)–(6).

$$P_{M(E_G,\phi)}(C_G|E_G) = P(C_G|E_G) \quad (7)$$

$$P_{M(E_P,\phi)}(C_G|E_G) = \sum_{\forall E_P} P(E_P|E_G) \times P(C_G|E_P) \quad (8)$$

$$P_{M(E_{GP},\phi)}(C_G|E_G) = \sum_{\forall E_P} P(E_P|E_G) \times P(C_G|E_G, E_P) \quad (9)$$

The three basic transliteration models in $\mathbf{M_I}$ are represented in Eqs. (7)–(9). Because the $\mathbf{M_I}$ models are independent of Chinese phonemes, they are the same as the transliteration models in the literature used for machine transliteration from English to other languages without relying on target-language phonemes (Karimi et al., 2007; Malik, 2006; Oh et al., 2006; Sherif and Kondrak, 2007; Yoon et al., 2007). Note that $M(E_G, \phi)$ is the same transliteration model as the one proposed by Li et al. (2004).

3.2 Hybrid Transliteration Models

The hybrid transliteration models in each class are defined by discrete mixture between the probability distribution of the two basic transliteration models, as in Eq. (10) (Al-Onaizan and Knight, 2002; Oh et al., 2006), where $0 < \alpha < 1$. We denote a hybrid transliteration model between two basic transliteration models $M(x_1, y)$ and $M(x_2, y)$ as $M(x_1 + x_2, y, \alpha)$, where $y \in Y = \{\phi, C_P, JC_P\}$, $x_1 \neq x_2$, and $x_1, x_2 \in X = \{E_G, E_P, E_{GP}\}$. In this paper, we define three types of hybrid transliteration models in each class: $M(E_G + E_P, y, \alpha)$, $M(E_G + E_{GP}, y, \alpha)$, and $M(E_P + E_{GP}, y, \alpha)$.

$$P_{M(x_1+x_2,y,\alpha)}(C_G|E_G) = \alpha \times P_{M(x_1,y)}(C_G|E_G) + (1-\alpha) \times P_{M(x_2,y)}(C_G|E_G) \quad (10)$$

3.3 Probability Estimation

Because Eqs. (1)–(9) can be estimated in a similar way, we limit our focus to Eq. (3) in this section. Assuming that $P(E_P|E_G)$, $P(C_P|E_G, E_P)$, and $P(C_G|E_G, E_P, C_P)$ in Eq. (3) depend on the size of the context window, k ($k = 3$ in this paper),

Table 3: Feature functions for $P(cg_i|cg_{i-k}^{i-1}, \langle eg, ep, cp\rangle_{i-k}^{i+k})$ with an example in Table 2, where $i = 2$

f_1	$gram_3(eg_i)$	eg_i^{i+2} = "ree"	cg_i = "里:B"
f_2	$pair_{11}(cp_{i-1}, cg_{i-1})$	cp_{i-1} = "G", cg_{i-1} = "格:B"	cg_i = "里:B"
f_3	$pair_{12}(cg_{i-1}, cp_{i-1})$	cp_{i-1}^i = "GE L", cg_{i-1}= "格:B"	cg_i = "里:B"
f_4	$pair_{22}(cp_{i-1}, cg_{i-2})$	eg_{i-1}^i = "gr", ep_{i-1}^i = "G R"	cg_i = "里:B"
f_5	$triple_1(eg_i, cp_i, cg_{i-1})$	eg_i = "r", cp_{i-1} = "GE", cg_{i-1}= "格:B"	cg_i = "里:B"
f_6	$triple_2(eg_{i-1}, cg_{i-1}, cp_{i-1})$	eg_{i-1} = "g", cp_{i-1}^i= "GE L", cg_{i-1}= "格:B"	cg_i = "里:B"

they can be simplified into a series of products in Eqs. (11)–(13).

The maximum entropy model is used to estimate the probabilities in Eqs. (11)–(13) (Berger et al., 1996). Generally, a conditional maximum entropy model is an exponential model that gives the conditional probability, as described in Eq. (14), where λ_i is the parameter to be estimated and $f_i(a, b)$ is a feature function corresponding to λ_i (Berger et al., 1996; Ratnaparkhi, 1997):

$$P(E_P|E_G) \approx \prod_i P(ep_i|ep_{i-k}^{i-1}, eg_{i-k}^{i+k}) \quad (11)$$

$$P(C_P|E_G, E_P) \quad (12)$$
$$\approx \prod_i P(cp_i|cp_{i-k}^{i-1}, \langle eg, ep\rangle_{i-k}^{i+k})$$

$$P(C_G|E_G, E_P, C_P) \quad (13)$$
$$\approx \prod_i P(cg_i|cg_{i-k}^{i-1}, \langle eg, ep, cp\rangle_{i-k}^{i+k})$$

$$P(b|a) = \frac{exp(\sum_i \lambda_i f_i(a, b))}{\sum_{b'} exp(\sum_i \lambda_i f_i(a, b'))} \quad (14)$$

$f_i(a, b)$ is a binary function returning TRUE or FALSE based on context a and output b. If $f_i(a, b)$=1, its corresponding model parameter λ_i contributes toward conditional probability $P(b|a)$ (Berger et al., 1996; Ratnaparkhi, 1997). The feature functions used here are defined in terms of context predicates — a function returning TRUE or FALSE that depends on the presence of the information in the current context (Ratnaparkhi, 1997). Context predicates and their descriptions used are given in Table 4.

N-GRAM includes $gram_1(u_j)$, $gram_2(u_j)$, and $gram_3(u_j)$ corresponding to a unigram, a bigram, and a trigram, respectively. PAIR includes a pair of unigrams ($pair_{11}$), unigram and bigram ($pair_{12}$), and bigrams ($pair_{22}$). TRIPLE includes a triple of three unigrams ($triple_1$) and a triple of two unigrams and one bigram ($triple_2$). Note that if different context predicates represent the same context, we accept one of them and ignore the others

Table 4: Context predicates and their descriptions

Category	Context predicates	Description
N-GRAM	$gram_1(u_j)$	u_j
	$gram_2(u_j)$	u_j^{j+1}
	$gram_3(u_j)$	u_j^{j+2}
PAIR	$pair_{11}(u_j, v_k)$	u_j, v_k
	$pair_{12}(u_j, v_k)$	u_j, v_k^{k+1}
	$pair_{22}(u_j, v_k)$	u_j^{j+1}, v_k^{k+1}
TRIPLE	$triple_1(u_j, v_k, w_l)$	u_j, v_k, w_l
	$triple_2(u_j, v_k, w_l)$	u_j, v_k, w_l^{l+1}

(e.g., $pair_{12}(u_j, u_{j+1}) = trigram(u_j) = u_j^{j+2}$). Table 3 represents the examples of feature functions for $P(cg_i|cg_{i-k}^{i-1}, \langle eg, ep, cp\rangle_{i-k}^{i+k})$.

We used the "Maximum Entropy Modeling Toolkit"[4] to estimate the probabilities and the LBFGS algorithm to find λ_i in Eq. (14). For each transliteration model, we produced n-best transliterations using a stack decoder (Schwartz and Chow, 1990).

3.4 Summary

In this paper, we defined eighteen transliteration models to be compared. There are six transliteration models, three basic and three hybrid ones, in each class, $\mathbf{M_I}$, $\mathbf{M_S}$, and $\mathbf{M_J}$. We compared the transliteration models from the viewpoint of Chinese phonemes or the class of transliteration models in our experiments.

4 Testing and Results

We used the same test set used in Li et al. (2004) for our testing[5]. It contains 37,694 pairs of English words and their official Chinese transliterations

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

[5]This test set was also used in "NEWS09 machine transliteration shared task" for English-to-Chinese transliteration (Li et al., 2009)

extracted from the "Chinese Transliteration of Foreign Personal Names" (Xinhua News Agency, 1992), which includes names in English, French, German, and many other foreign languages (Li et al., 2004). We used the same test data as in Li et al. (2004). But we randomly selected 90% of the training data used in Li et al. (2004) as our training data and the remainder as the development data, as shown in Table 5.

Table 5: Number of English-Chinese transliteration pairs in each data set

	Ours	Li et al. (2004)
Training data	31,299	34,777
Development data	3,478	N/A
Blind test data	2,896	2,896

We used the training data for training the transliteration models. For each model, we tuned the parameters including the number of iterations for training the maximum entropy model and a Gaussian prior for smoothing the maximum entropy model using the development data. Further, the development data was used to select parameter α of the hybrid transliteration models. We varied parameter α from 0 to 1 in 0.1 intervals (i.e., α=0, 0.1, 0.2, $\cdots$,1) and tested the performance of the hybrid models with the development data. Then we chose α that showed the best performance in each hybrid model. The blind test data was used for evaluating the performance of each transliteration model. *The CMU Pronouncing Dictionary*[6], which contains about 120,000 English words and their pronunciations, was used for estimating $P(E_P|E_G)$.

We conducted two experiments. First, we compared the overall performance of the transliteration models. Second, we investigated the effect of training data size on the performance of each transliteration model.

The evaluation was done for word accuracy in top-1 (ACC), Chinese pronunciation accuracy (CPA) and a mean reciprocal rank (MRR) metric (Kantor and Voorhees, 2000; Li et al., 2009; Chang et al., 2009). ACC measures how many correct transliterations appeared in the top-1 result of each system. CPA measures the Chinese pronunciation accuracy in the top-1 of the n-best Chinese pronunciation. We used CPA for comparing the performance between systems based on Chinese phonemes. MRR, mean reciprocal ranks of n-best results of each system over the test entries, is an evaluation measure for n-best transliterations. If a transliteration generated by a system matches a reference transliteration[7] at the r^{th} position of the n-best results, its reciprocal rank equals $1/r$; otherwise its reciprocal rank equals 0, where $1 \leq r \leq n$. We produced 10-best Chinese transliterations for each English word in our experiments.

4.1 Comparison of the Overall Performance

Table 6 represents the overall performance of one system in a previous work (Li et al., 2004) and eighteen systems based on the transliteration models defined in this paper. ACC, MRR, and CPA represent the evaluation results for each model trained by our training data. To test transliteration models without the errors introduced by incorrect Chinese phonemes, we carried out the experiments with the correct Chinese pronunciation (or the correct Chinese phoneme sequence) in Chinese phoneme-to-grapheme conversion. In the experiment, we put the correct Chinese pronunciation into the top-1 of the n-best Chinese pronunciation with the highest probability, say $P(C_P|E_G)$=1; thus CPA was assumed to be 100%. The ACC of the transliteration models under this condition is denoted as ACC' in Table 6. TRAIN represents the evaluation results of the transliteration models trained by our training data. To compare Li et al. (2004) and transliteration models defined in this paper under the same condition, we also carried out experiments with the same training data in Li et al. (2004). Since the training data used in Li et al. (2004) is identical as the union of our training and development data, we denoted it as TRAIN+DEV in Table 6. In both TRAIN and TRAIN+DEV, we used the same parameter setting that was obtained by using the development data.

LI04 represents a system in Li et al. (2004), and its ACC' in TRAIN+DEV is taken from the literature. The systems based on the transliteration models defined in our paper are represented from the second row in Table 6. The phoneme-based transliteration models in the literature correspond to either $M(E_G, C_P)$ (Wan and Verspoor, 1998; Lee and Chang, 2003; Jiang et al., 2007) or $M(E_P, C_P)$ (Meng et al., 2001; Gao et al., 2004;

[6]Available at `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`

[7]In our test set, an English word corresponds to one reference Chinese transliteration.

Table 6: Comparison of the overall performance

Class	Model	TRAIN				TRAIN+DEV			
		ACC	MRR	CPA	ACC'	ACC	MRR	CPA	ACC'
LI04		N/A	N/A	N/A	N/A	70.1	N/A	N/A	N/A
$\mathbf{M_J}$	$M(E_G, JC_P)$	71.9	80.4	72.3	88.2	72.3	80.7	73.1	88.9
	$M(E_P, JC_P)$	61.1	70.3	62.4	82.8	61.1	70.6	63.1	83.8
	$M(E_{GP}, JC_P)$	72.3	80.9	73.2	89.6	73.5	81.5	73.9	90.4
	$M(E_G{+}E_P, JC_P, 0.7)$	72.8	80.7	73.8	89.7	73.2	81.0	74.7	90.5
	$M(E_G{+}E_{GP}, JC_P, 0.6)$	**73.5**	**81.7**	**74.2**	**90.6**	**73.7**	**81.8**	**74.8**	**91.2**
	$M(E_P{+}E_{GP}, JC_P, 0.1)$	71.6	80.3	73.3	89.8	72.5	80.8	73.8	90.1
$\mathbf{M_I}$	$M(E_G, \phi)$	70.0	78.5	N/A	N/A	70.6	79.0	N/A	N/A
	$M(E_P, \phi)$	58.5	69.3	N/A	N/A	59.4	70.1	N/A	N/A
	$M(E_{GP}, \phi)$	71.2	79.9	N/A	N/A	72.3	80.7	N/A	N/A
	$M(E_G{+}E_P, \phi, 0.7)$	70.7	79.1	N/A	N/A	72.0	80.0	N/A	N/A
	$M(E_G{+}E_{GP}, \phi, 0.4)$	72.0	80.3	N/A	N/A	72.8	80.9	N/A	N/A
	$M(E_P{+}E_{GP}, \phi, 0.1)$	71.0	79.6	N/A	N/A	72.0	80.4	N/A	N/A
$\mathbf{M_S}$	$M(E_G, C_P)$	58.9	70.2	72.3	78.4	59.1	70.4	73.1	78.4
	$M(E_P, C_P)$	50.2	62.3	62.4	78.4	50.4	62.6	63.1	78.5
	$M(E_{GP}, C_P)$	59.1	70.4	73.2	78.4	59.3	70.5	73.9	78.5
	$M(E_G{+}E_P, C_P, 0.8)$	59.7	71.3	73.8	79.0	60.3	71.7	74.7	79.0
	$M(E_G{+}E_{GP}, C_P, 0.6)$	59.8	71.7	74.2	78.9	60.6	72.1	74.8	78.9
	$M(E_P{+}E_{GP}, C_P, 0.1)$	58.8	70.4	73.3	78.9	59.4	70.7	73.8	78.8

Virga and Khudanpur, 2003).

A comparison between the basic and hybrid transliteration models showed that the hybrid ones usually performed better (the exception was $M(E_P{+}E_{GP}, y, \alpha)$ but the performance still comparable to the basic ones in each class). Especially, the hybrid ones based on the best two basic transliteration models, $M(E_G{+}E_{GP}, y, \alpha)$, showed the best performance.

A comparison among the $\mathbf{M_I}$, $\mathbf{M_S}$, and $\mathbf{M_J}$ models showed that Chinese phonemes did contribute to the performance improvement of English-to-Chinese transliteration when Chinese phonemes were used together with their corresponding English graphemes and phonemes in Chinese phoneme-to-grapheme conversion. A one-tail paired t-test between the $\mathbf{M_I}$ and $\mathbf{M_J}$ models showed that the results of the $\mathbf{M_J}$ models were always significantly better than those of the $\mathbf{M_I}$ models if the $\mathbf{M_I}$ and $\mathbf{M_J}$ models shared the same English-side parameter, $x \in \{E_G, E_P, E_{GP}\}$ (level of significance = 0.001). In the results obtained by the $\mathbf{M_S}$ and $\mathbf{M_J}$ models, the figures in CPA are the same when the $\mathbf{M_S}$ and our $\mathbf{M_J}$ models share the same English-side parameter. Moreover, the difference between the figures in ACC and CPA can be interpreted as the error rate of Chinese phoneme-to-grapheme conversion. Our proposed $\mathbf{M_J}$ models generated Chinese transliterations with a very low error rate in Chinese phoneme-to-grapheme conversion, while the $\mathbf{M_S}$ models suffered from a significant error rate in Chinese phoneme-to-grapheme conversion. ACC' showed that the $\mathbf{M_J}$ models still outperformed the $\mathbf{M_S}$ models even without errors in generating Chinese pronunciation from the English words. These results indicate that the joint use of Chinese phonemes and their corresponding English graphemes and phonemes significantly improved the performance in Chinese phoneme-to-grapheme conversion and English-to-Chinese transliteration.

Table 7 shows the Chinese transliterations generated by $M(E_G, \phi)$, $M(E_{GP}, \phi)$, $M(E_G, JC_P)$, and $M(E_{GP}, JC_P)$ where English or Chinese phonemes contributed to the correct transliteration. In this table, the first column show the English words and their English phonemes, and the second and third columns represent the Chinese transliterations and their phonemes. Note that the Chinese phonemes in the second and third columns of the $\mathbf{M_I}$ models are not used in transliteration. They are shown in the table to indicate the difference in the Chinese phonemes of Chinese

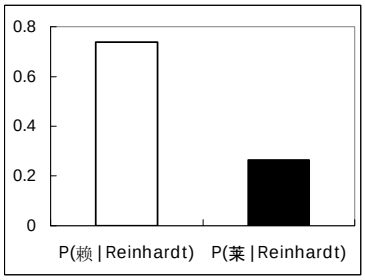
Table 7: Top-1 results of $M(E_G, \phi)$, $M(E_{GP}, \phi)$, $M(E_G, JC_P)$, and $M(E_{GP}, JC_P)$, where * represents incorrect transliterations

$\mathbf{M_I}$ models	$M(E_G, \phi)$	$M(E_{GP}, \phi)$
Emily (EH M IH L IY)	埃米利* (AI MI LI)	埃米利* (AI MI LI)
Ivy (AY V IY)	伊维* (YI WEI)	艾维 (AI WEI)
Reinhardt (R AI N HH AA R T)	赖因哈特* (LAI YIN HA TE)	赖因哈特* (LAI YIN HA TE)
$\mathbf{M_J}$ models	$M(E_G, JC_P)$	$M(E_{GP}, JC_P)$
Emily (EH M IH L IY)	埃米莉 AI MI LI	埃米莉 AI MI LI
Ivy (AY V IY)	伊维* YI WEI	艾维 AI WEI
Reinhardt (R AI N HH AA R T)	莱因哈特 LAI YIN HA TE	莱因哈特 LAI YIN HA TE

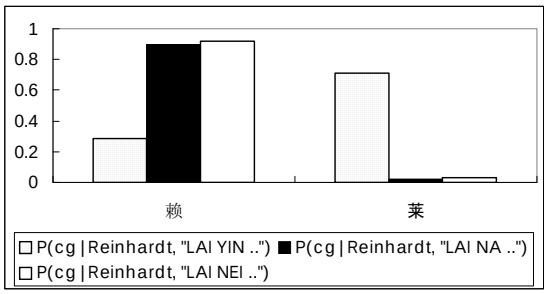
transliterations between the $\mathbf{M_I}$ and $\mathbf{M_J}$ models.

For *Emily* and *Reinhardt*, the $\mathbf{M_J}$ models generated correct Chinese transliterations, but the $\mathbf{M_I}$ models did not. Figure 1 shows the probability distribution when a transliteration model generates the first Chinese character in the Chinese transliteration of *Reinhardt* with and without Chinese phonemes. Two Chinese characters, 赖 and 莱, were strong candidates and 莱 is the correct one in this case. Without Chinese phonemes, $M(E_G, \phi)$, which is based on P(cg|$Reinhardt$) in Figure 1(a) preferring 赖 to 莱, generated the incorrect transliteration as shown in Table 7. However, Figure 1(b) shows that 莱 can be selected if the correct Chinese phoneme sequence "LAI YIN ..." is given. Three Chinese phoneme sequences starting with "LAI YIN ...", "LAI NA ...", and "LAI NEI ..." were generated from *Reinhardt*, where "LAI YIN ..." was the best Chinese phoneme sequence based on the probability distribution in Figure 1(c). As a result, $M(E_G, JC_P)$, which jointly used Chinese phonemes with English graphemes, generated the correct Chinese transliteration of *Reinhardt* based on two probability distribution in Figures 1(b) and 1(c). In the case of *Ivy*, English phonemes contributed to generating the correct transliteration in the $M(E_{GP}, \phi)$ and $M(E_{GP}, JC_P)$ models.

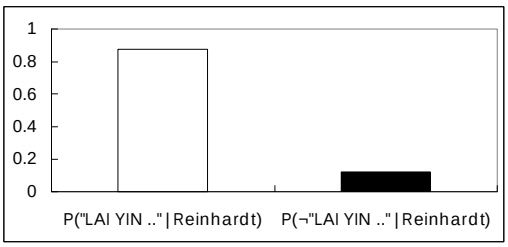
Chinese transliterations sometimes reflect the English word's pronunciation as well as the Chinese character's meaning (Li et al., 2007). Li et al. (2007) defined such a Chinese transliteration as a phonetic-semantic transliteration (semantic transliteration) to distinguish it from a usual phonetic transliteration. One fact that affects semantic transliteration is gender association (Li et al., 2007). For example, 莉 (meaning *jasmine*) is frequently used in Chinese transliterations of female names but seldom in common person names. Because *Emily* is often used in female names, the results obtained by the $M(E_G, JC_P)$ and $M(E_{GP}, JC_P)$ models are acceptable. This indicates that Chinese phonemes coupled with English graphemes or those coupled with English graphemes and phonemes could provide evidence required for semantic transliteration as well as phonetic transliteration. As a result, $M(E_{GP}, \phi)$, $M(E_G, JC_P)$,



(a) Probability distribution when Chinese phonemes are not given



(b) Probability distribution when Chinese phonemes are given

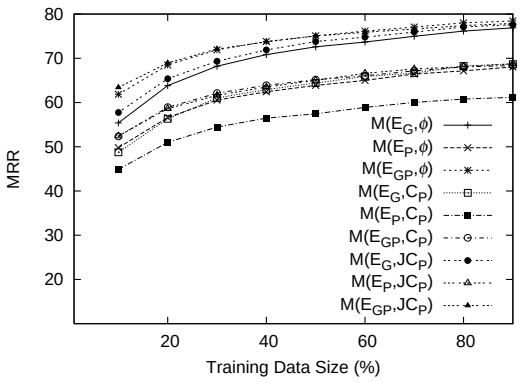


(c) Probability distribution for Chinese phoneme sequence "LAI YIN ..." and others

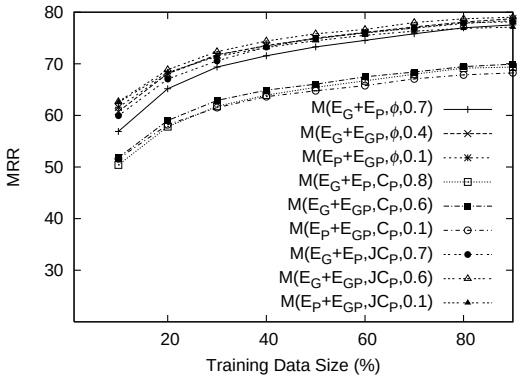
Figure 1: Probability distribution for the first Chinese character in the Chinese transliteration of *Reinhardt*: $M(E_G, \phi)$ vs. $M(E_G, JC_P)$

and $M(E_{GP}, JC_P)$, which used phonemes coupled with English graphemes, achieved higher performance than $M(E_G, \phi)$, which relied only on English graphemes.

4.2 Effect of Training Data Size



(a) Basic transliteration models



(b) Hybrid transliteration models

Figure 2: Performance of each system with different training data size

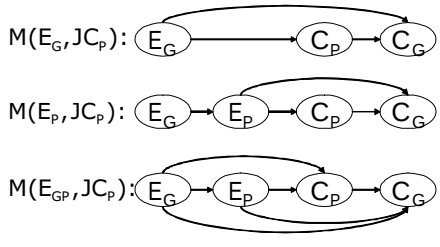
We investigated the effect of training data size on the performance of each transliteration model. We randomly selected training data with ratios from 10 to 90% and compared the performance of each system trained by different sizes of training data. The results for the basic transliteration models in Figure 2(a) can be categorized into three groups. $M(E_{GP}, \phi)$ and $M(E_{GP}, JC_P)$ fall into the best group, where they showed the best performance regardless of training data size. $M(E_G, \phi)$ and $M(E_G, JC_P)$ belong to the middle group, where they showed lower performance than the best group if the training data size is small, but their performance is comparable to the best group if the size of the training data is large enough. The others always showed lower performance than both the best and middle groups. Figure 2(b) shows that hybrid transliteration models, on average, were less sensitive to the training data size than the basic ones, because the two different basic transliteration models used in the hybrid ones boosted transliteration performance by complementing each other's weak points.

5 Conclusion

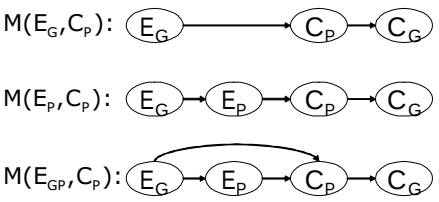
We proposed a new English-to-Chinese transliteration model based on Chinese phonemes and their corresponding English graphemes and phonemes. We defined eighteen English-to-Chinese transliteration models including our proposed model and classified them into three classes based on the role of Chinese phonemes in the transliteration models. Experiments showed that Chinese phonemes in our proposed model can contribute to the performance improvement in English-to-Chinese transliteration.

Now we can answer *Yes* to this paper's key question, "Can Chinese phonemes improve machine transliteration?" Actually, this is the second time the same question has been answered. The previous answer, which was unfortunately reported as *No* by Li et al. (2004), has been accepted as true for the last five years; the research issue has been considered closed. In this paper, we found a new answer that contradicts the previous answer. We hope that our answer promotes research on phoneme-based English-to-Chinese transliteration.

Appendix: Illustration of Basic Transliteration Models in $\mathbf{M_J}$ and $\mathbf{M_S}$



(a) $\mathbf{M_J}$ models



(b) $\mathbf{M_S}$ models

References

Y. Al-Onaizan and Kevin Knight. 2002. Translating named entities using monolingual and bilingual resources. In *Proc. of ACL '02*, pages 400–408.

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

M. Chang, D. Goldwasser, D. Roth, and Y. Tu. 2009. Unsupervised constraint driven learning for transliteration discovery. In *Proceedings of NAACL HLT' 09*.

Wei Gao, Kam-Fai Wong, and Wai Lam. 2004. Phoneme-based transliteration of foreign names for OOV problem. In *Proc. of IJCNLP 2004*, pages 110–119.

Long Jiang, Ming Zhou, Lee-Feng Chien, and Cheng Niu. 2007. Named entity translation with web mining and transliteration. In *Proc. of IJCAI '07*, pages 1629–1634.

Paul B. Kantor and Ellen M. Voorhees. 2000. The trec-5 confusion track: Comparing retrieval methods for scanned text. *Information Retrieval*, 2:165–176.

Sarvnaz Karimi, Falk Scholer, and Andrew Turpin. 2007. Collapsed consonant and vowel models: New approaches for English-Persian transliteration and back-transliteration. In *Proceedings of ACL '07*, pages 648–655.

Chun-Jen Lee and Jason S. Chang. 2003. Acquisition of English-Chinese transliterated word pairs from parallel-aligned texts using a statistical machine transliteration model. In *Proc. of HLT-NAACL 2003 Workshop on Building and Using Parallel Texts*, pages 96–103.

Haizhou Li, Min Zhang, and Su Jian. 2004. A joint source-channel model for machine transliteration. In *Proceedings of the 42th Annual Meeting of the Association of Computational Linguistics*, pages 160–167.

Haizhou Li, Khe Chai Sim, Jin-Shea Kuo, and Minghui Dong. 2007. Semantic transliteration of personal names. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

Haizhou Li, A Kumaran, Min Zhang, and Vladimir Pervouchine. 2009. Whitepaper of NEWS 2009 machine transliteration shared task. In *Proc. of ACL-IJCNLP 2009 Named Entities Workshop*.

M.G. Abbas Malik. 2006. Punjabi machine transliteration. In *Proceedings of the COLING/ACL 2006*, pages 1137–1144.

H.M. Meng, Wai-Kit Lo, Berlin Chen, and K. Tang. 2001. Generating phonetic cognates to handle named entities in English-Chinese cross-language spoken document retrieval. In *Proc. of Automatic Speech Recognition and Understanding, 2001. ASRU '01*, pages 311–314.

Jong-Hoon Oh, Key-Sun Choi, and Hitoshi Isahara. 2006. A comparison of different machine transliteration models. *Journal of Artificial Intelligence Research (JAIR)*, 27:119–151.

Adwait Ratnaparkhi. 1997. A linear observed time statistical parser based on maximal entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 1–10.

Richard Schwartz and Yen-Lu Chow. 1990. The N-Best algorithm: an efficient procedure for finding top N sentence hypotheses. In *Proc. of ICASSP '90*, pages 81–84.

Tarek Sherif and Grzegorz Kondrak. 2007. Substring-based transliteration. In *Proceedings of ACL '07*, pages 944–951.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-lingual information retrieval. In *Proc. of ACL 2003 Workshop on Multilingual and Mixed-language Named Entity Recognition*, pages 57–64.

Stephen Wan and Cornelia Maria Verspoor. 1998. Automatic English-Chinese name transliteration for development of multilingual resources. In *Proc. of COLING '98*, pages 1352–1356.

Xinhua News Agency. 1992. *Chinese transliteration of foreign personal names*. The Commercial Press.

Binyong Yin and Mary Felley. 1990. *Chinese Romanization: Pronunciation and Orthography*. Sinolingua.

Su-Youn Yoon, Kyoung-Young Kim, and Richard Sproat. 2007. Multilingual transliteration using feature based phonetic method. In *Proceedings of ACL'07*, pages 112–119.

Unsupervised morphological segmentation and clustering with document boundaries

Taesun Moon, Katrin Erk, and Jason Baldridge
Department of Linguistics
University of Texas at Austin
1 University Station B5100
Austin, TX 78712-0198 USA
{tsmoon,katrin.erk,jbaldrid}@mail.utexas.edu

Abstract

Many approaches to unsupervised morphology acquisition incorporate the frequency of character sequences with respect to each other to identify word stems and affixes. This typically involves heuristic search procedures and calibrating multiple arbitrary thresholds. We present a simple approach that uses no thresholds other than those involved in standard application of χ^2 significance testing. A key part of our approach is using document boundaries to constrain generation of candidate stems and affixes and clustering morphological variants of a given word stem. We evaluate our model on English and the Mayan language Uspanteko; it compares favorably to two benchmark systems which use considerably more complex strategies and rely more on experimentally chosen threshold values.

1 Introduction

Unsupervised morphology acquisition attempts to learn from raw corpora one or more of the following about the *written* morphology of a language: (1) the segmentation of the set of word types in a corpus (Creutz and Lagus, 2007), (2) the clustering of word types in a corpus based on some notion of morphological relatedness (Schone and Jurafsky, 2000), (3) the generation of out-of-vocabulary items which are morphologically related to other word types in the corpus (Yarowsky et al., 2001).

We take a novel approach to segmenting words and clustering morphologically related words. The approach uses no parameters that need to be tuned on data. The two main ideas of the approach are (a) the filtering of affixes by significant co-occurrence, and (b) the integration of knowledge of document boundaries when generating candidate stems and affixes and when clustering morphologically related words. The main application that we envision for our approach is to produce interlinearized glossed texts for under-resourced/endangered languages (Palmer et al., 2009). Thus, we strive to eliminate hand-tuned parameters to enable documentary linguists to use our model as a preprocessing step for their manual analysis of stems and affixes. To require a documentary linguist–who is likely to have little to no knowledge of NLP methods–to tune parameters is unfeasible. Additionally, data-driven exploration of parameter settings is unlikely to be reliable in language documentation since datasets typically are quite small. To be relevant in this context, a model needs to produce useful results out of the box.

Constraining learning by using document boundaries has been used quite effectively in unsupervised word sense disambiguation (Yarowsky, 1995). Many applications in information retrieval are built on the statistical correlation between documents and terms. However, we are unaware of cases where knowledge of document boundaries has been used for unsupervised learning for morphology. The intuition behind our approach is very simple: if two words in a single document are very similar in terms of orthography, then the two words are likely to be related morphologically. We measure how integrating these assumptions into our model at different stages affects performance.

We define a simple pipeline model. After generating candidate stems and affixes (possibly constrained by document boundaries), a χ^2 test based on global corpus counts filters out unlikely affixes. Mutually consistent affix pairs are then clustered to form affix groups. These in turn are used to build morphologically related word clusters, possibly constrained by evidence from co-occurence of word forms in documents. Following Schone and Jurafsky (2000), clusters are evaluated for

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 668–677,
Singapore, 6-7 August 2009.

whether they capture inflectional paradigms using CELEX (Baayen et al., 1993).

We are unaware of other work on morphology using χ^2 tests despite its wide application across many disciplines.[1] This may be due to the large degree of noise found in the candidate affix sets induced through other candidate generation methods. The χ^2 test has two standard thresholds–a significance threshold and a lower bound on observed counts. These are the only manually set parameters we require—and we in fact use the widely accepted standard values for these thresholds without varying them in our experiments. This is a significant improvement over other approaches that typically require a number of arbitrary thresholds and parameters yet provide little intuitive justification for them. (We give examples of these in §3.)

We evaluate our approach on two languages, English and Uspanteko, and compare its performance to two benchmark systems, Morfessor (Creutz and Lagus, 2007) and Linguistica (Goldsmith, 2001). English is commonly used in other studies and permits the use of CELEX as a gold standard for evaluation. Uspanteko is an endangered Mayan language for which we have a set of interlinearized glossed texts (IGT) (Pixabaj et al., 2007; Palmer et al., 2009). IGT provides word-by-word morpheme segmenation, which we use to create a synthetic gold standard. In addition to evaluation against this standard, Telma Kaan Pixabaj—a Mayan linguist who helped create the annotated corpus—reviewed by hand 100 word clusters produced by our system, Morfessor and Linguistica. Note that because English is suffixal and Uspanteko is both prefixal and suffixal, we use a slightly modified model for Uspanteko.

The approach introduced in this paper compares favorably to Linguistica and Morfessor, two models that employ much more complex strategies and rely on experimentally-tuned language/corpus-specific parameters. In our evaluation, document boundary awareness greatly benefits precision for small datasets, blocking acquisition of spurious affixes. For large datasets, global candidate generation outperforms document-aware candidate generation at the task of filtering out spurious stems, but document-aware clustering improves precision. These findings are promising for the application of this approach to under-resourced languages like Uspanteko.

[1] Monson (2004) suggests, but does not actually use, χ^2.

2 Unsupervised morphology acquisition

Unsupervised morphology acquisition aims to model one or more of three properties of *written* morphology: segmentation, clustering around a common stem, and generation of new word forms with productive affixes. Intuitively, there are straightforward, but non-trivial, challenges that arise when evaluating a model. One large challenge is distinguishing derivational from inflectional morphology. Most approaches deal with tokens without considering context. Since inflectional morphology is virtually always driven by syntax and word context, such approaches are unable to learn only inflectional morphology or only derivational morphology. Even approaches which take context into consideration (Schone and Jurafsky, 2000; Baroni et al., 2002; Freitag, 2005) cannot learn specifically for one or the other.

In addition, the evaluation of both segmentation and clustering involves arbitrary judgment calls. Concerning segmentation, should *altimeter* and *altitude* be one morpheme or two? (The sample English gold standard for MorphoChallenge 2009 provides *alti+meter* but *altitude*.) Similar issues arise when evaluating clusters of related word forms if inflection and derivation are not distinguished. Does *atheism* belong to the same cluster as *theism*? Where is the frequency cutoff point between a productive derivational morpheme and an unproductive one? Yet, many studies have evaluated their segmentations and clusters by going over their results word by word, cluster by cluster and judging by sight whether some segmentation or clustering is good (e.g., Goldsmith (2001)).

Like Schone and Jurafsky (2001), we build clusters that will have both inflectionally and derivationally related stems and evaluate them with respect to a gold standard of *only* inflectionally related stems.

3 Related work

There is a diverse body of existing work on unsupervised morphology acquisition. We summarize previous work, emphasizing some of its more arbitrary and *ad hoc* aspects.

Letter successor variety. Letter successor variety (LSV) models (Hafer and Weiss, 1974; Gaussier, 1999; Bernhard, 2005; Bordag, 2005;

Keshava and Pitler, 2005; Hammarström, 2006; Dasgupta and Ng, 2007; Demberg, 2007) use the hypothesis that there is less certainty when predicting the next character at morpheme boundaries. LSV has several issues that require fine parameter tuning. For example, Hafer and Weiss (1974) counts how many types of characters appear after some initial string (the *successor* count) and how many types of characters appear before some final string (the *predecessor* count). A successful criterion for segmenting a word was if the predecessor count for the second part was greater than 17 and the successor count for the first part was greater than 5. Other studies have similar data specific parameters and restrictions.

MDL and Bayesian models. Minimum description length (MDL) models (Goldsmith, 2001; Creutz and Lagus, 2002; Creutz and Lagus, 2004; Goldsmith, 2006; Creutz and Lagus, 2007) try to segment words by maximizing the probability of a training corpus subject to a penalty based on the size of hypothesized morpheme lexicons they build on the basis of the segmentations. While theoretically elegant, a pure implementation on real data results in descriptions that do not reflect actual morphology. Creutz and Lagus (2005) report that, "frequent word forms remain unsplit, whereas rare word forms are excessively split." In the end, every MDL approach uses probabilistically motivated refinements that restrict the tendency of raw MDL to generate descriptions that do not fit linguistic notions of morphology. Despite the sophistication of the models in this group, there are many parameters that need to be set, and heuristic search procedures are crucial for their success (Goldwater, 2007). Snover et al. (2002) present a Bayesian model that uses a prior distribution to refine disjoint clusters of morphologically related words. It disposes with parameter setting by selecting the highest ranking hypothesis.

Context aware approaches. A word's morphology is strongly influenced by its syntactic and semantic context. Schone and Jurafsky (2000) attempts to cluster morphologically related words starting with an unrefined trie search (but with a parameter of minimum possible stem length and an upper bound on potential affix candidates) that is constrained by semantic similarity in a word context vector space. Schone and Jurafsky (2001) builds on this approach, but adds more *ad hoc* parameters to handle circumfixation. Baroni et al. (2002) takes a similar approach but uses edit distance to cluster words that are similar but do not necessarily share a long, contiguous substring. They remove noise by constraining cluster membership with mutual information derived semantic similarity. Freitag (2005) uses a mutual information derived measure to learn the *syntactic* similarity between words and clusters them. Then he derives finite state machines across words in different clusters and refines them through a graph walk algorithm. This group is the only one to evaluate against CELEX (Schone and Jurafsky, 2000; Schone and Jurafsky, 2001; Freitag, 2005).

Others. Some other models require input such as POS tables and lexicons and use a wider range of information about the corpus (Yarowsky and Wicentowski, 2000; Yarowsky et al., 2001; Chan, 2006). Because of the knowledge dependence of these models, they are able to properly induce inflectional morphology, as opposed to the studies cited above. Snyder and Barzilay (2008) uses a set of aligned phrases across related languages to learn how to segment words with a Bayesian model and is otherwise fully unsupervised.

4 Model[2]

Our goal is to generate *conflation sets*: sets of word types that are related through either inflectional or derivational morphology (Schone and Jurafsky, 2000). Solving this task requires learning how individual types are segmented (though the segmentation itself is not evaluated). For present purposes, we assume that the affixal pattern of the language is known: whether it is prefixal, suffixal, or both. To simplify presentation, we discuss a model that captures suffixes only. Our approach is a four stage process:

1. *Candidate Generation*: generate candidate stems and affixes using an orthographically defined data structure (a trie)

2. *Candidate Filtering*: filter candidate affixes using the statistical significance for pairs of affixes based on their co-occurence counts with shared stems

3. *Affix Clustering*: cluster significant affix pairs into affix groups

[2]The code implementing the model is available from `http://comp.ling.utexas.edu/earl`

4. *Word Clustering*: form conflation sets based on affix clusters

The first and last stages are particularly prone to noise, which has necessitated many of the thresholds and heuristics employed in previous work. We hypothesize that naturally occuring document boundaries provide a strong constraint that should reduce this noise, and we test that hypothesis by using it in those stages.

Our intuition comes from an observation by Yarowsky (1995) regarding multiple tokens of words in documents. He tabulates the *applicability* of using document boundaries to disambiguate word senses, which measures how often a given word occurs more than twice in the same document. For ten potentially ambiguous words, he counts how often they occur more than once in some document and finds that if the words do occur, they do so multiple times in 50.1% of these documents, on average. His counts ignored morphological variation, and it is likely the *applicability* measure would have increased considerably: if a content word is used more than once in some text, it is likely to be repeated in different syntactic contexts, requiring the word to be inflected or to be derived for a different part-of-speech category. [3]

For stage one, we build separate tries for each document rather than a trie for the entire corpus. This should reduce the chance that orthographically similar but morphologically unrelated word pairs lead to bad candidates by reducing the search space for words which share a stem to a local document. For example, *assuage* and *assume* are both likely to occur in a large corpus and suggest that there is a stem *assu* with affixes *-age* and *-me*. They are less likely to occur together in many different documents that form the corpus, whereas *assume*, *assumed*, and *assuming* are. We refer to this document constrained candidate generation as *CandGen-D*, and to the unconstrained generation (a single trie for all documents) as *CandGen-G*.

For stage four, documents are used to constrain potential membership of words in clusters: all pairs of words in a cluster must have occured together in some document. We refer to document-constrained clustering as *Clust-D* and the unconstrained global clustering as *Clust-G*.

[3]For example, in just this *one* paragraph we have {*document,documents*}, {*measure, measures*}, {*occur, occurs, occuring*}, and {*word, words*}.

4.1 Candidate generation

Given a document or collection of documents, we use tries (prefix trees) to identify potential stems and affixes and collect statistics for co-occurrences between affixes and between affixes and stems. A trie G, like the example on the right, can be identified with the set of all words on paths from the root to any leaf, in the case of the example figure the set $G = \{abd, ab\$, ac\}$. (We use \$ to denote an empty affix.) Given a trie G over alphabet L, we define the set of *trunks* of G as all paths from the root to a branching point:

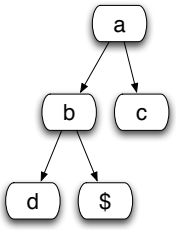


Figure 1

$$\mathit{Tr}(G) = \{w \in L^+ \mid \exists a, b \in L, x_1, x_2 \in L^* : a \neq b \ \wedge \ wax_1, wbx_2 \in G\}$$

Also, we define the set of *branches* of a trunk $t \in \mathit{Tr}(G)$ as the paths from its branching points to the leaves:

$$\mathit{Br}(t, G) = \{x \in L^+ \mid tx \in G\}$$

In our example, $\{a, ab\}$ are the trunks, with $\mathit{Br}(\mathrm{a}, G) = \{bd, b\$, c\}$ and $\mathit{Br}(\mathrm{ab}, G) = \{d, \$\}$. When we use a trie to induce stems and affixes, all induced stems will be trunks, and all induced affixes will be branches.

From a given trie, we induce a set of *stem candidates* and *affix candidates*. A simple criterion is used: if a trunk is longer than all of its branches, the trunk is a stem candidate and its branches are affix candidates. So, the set of stem candidates for a trie G, $\mathit{CStem}(G)$, is the set of trunks $t \in \mathit{Tr}(G)$ such that $|t| > |b|$ for all $b \in \mathit{Br}(t, G)$.

Given a stem candidate $s \in \mathit{CStem}(G)$, its set of affix candidates $\mathit{CAff}(s, G)$ is identical to its set of branches. (To talk about the sets of stem and affix candidates for a whole trie G or a set of tries, we write $\mathit{CAff}(G)$, $StC(G)$, CAff, and CStem.) The *count* of an affix candidate $b \in \mathit{CAff}$ is the number of stem candidates with which it occurs:

$$\mathrm{count}(b) = \sum_G |\{s \in \mathit{CStem}(G) \mid b \in \mathit{CAff}(s, G)\}|$$

For Fig. 1, the set of stem candidates is $\{ab\}$ (since some branches of the trunk a are longer than the

trunk itself). The matching set of affix candidates is *CAff*(ab, G) = $\{d, \$\}$, each with a count of one.

An *affix rule candidate* is an unordered pair of affix candidates $\{b_1, b_2\}$. It states that any stem occurring with b_1 can also occur with b_2. Affix rules implement the assumption that all productive affixes will cooccur with other productive affixes and that these will form a coherent group. The rule candidates for a given stem candidate $s \in$ *CStem*(G) are:

$$\textit{CRule}(s, G) = \left\{ \{b_1, b_2\} \subseteq \textit{CAff}(s, G) \mid b_1 \neq b_2 \right\}$$

For example, the single stem candidate *ab* in Fig. 1 has one rule candidate, $\{d, \$\}$. We also use *CRule*($G$) for the rule candidates of a trie G across all stems, and *CRule* for the union of rule candidates in a set of tries.

The count of a rule candidate r=$\{b_1, b_2\}$ in a trie is the number of stem candidates it appears with:

$$\text{count}(r) = \sum_{G} |\{s \in \textit{CStem}(G) \mid r \in \textit{CRule}(s, G)\}|$$

We also use *CAff*(s) for the set of affix candidates of stem s across several tries, and *CRule*(s) for the set of rule candidates of a stem s across several tries.

Document-specific versus global candidate generation. *CandGen-D* defines separate tries for every document in the corpus and induces stem, affix and rule candidates for each document. *CandGen-G* instead induces these candidates for a global trie over all the words in the corpus. From the perspective of the formalism laid out above, the only difference is that *CandGen-D* has as many tries G_i as there are documents i and *CandGen-G* has only one G. This simple difference leads to different candidate sets and counts over their occurrences. For example, say two documents contain the pair *putt/putts* and another contains *bogey/bogeys*. With *CandGen-D*, count($\$$)=3, count(s)=3, and count($\$, s$)=2. For the same documents, *CandGen-G* would produce count($\$$)=2 and count(s)=2 since *putt/putts* would have occurred only once in the global trie.

Also, consider a rare pair such as *aardvark/aardvarks* where each word is found in a different document. The pair would be identified by *CandGen-G* but not by *CandGen-D*. The pair would contribute a count of one to count($\$, s$) in *CandGen-G* but not in *CandGen-D*. So, *CandGen-G* can provide better coverage, but it is also more likely to identify noisy candidates, such as *assuage/assumed*, than *CandGen-D*.

4.2 Candidate filtering

The sets of candidates *CStem*, *CAff*, *CRule* is expected to be noisy since the only basis for generating them was strings that share a large portion of their substrings. One way of filtering candidates is to find affix candidates whose co-occurence with other candidates is not statistically significant.

We measure correlation between candidate affixes b_1, b_2 in a candidate rule with the paired χ^2 test. By using χ^2, we only consider pairwise correlation between affixes, rather than attempting global inference. Global consistency of affix sets is not ensured, and as such the approach is susceptible to the multiple comparisons problem. We still opt for this approach for its simplicity and because global inference is problematic due to data sparseness.

Correlation between b_1 and b_2 is determined by the following contingency table:[4]

	b_1	$\sim b_1$
b_2	O_{11}	O_{12}
$\sim b_2$	O_{21}	O_{22}

Based on the significance testing, we define the set of valid rules *PairRule* as those for which the χ^2 test is significant at $p < 0.05$. Thus, affix candidates not significantly correlated with any other affix in *CAff* are discarded.

4.3 Affix clustering

The previous stage produces a set of *pairs* of affixes that are significantly correlated. However, inflectional paradigms rarely contain just two affixes, so we would like to group together affix pairs into larger affix sets to improve generalization. We use a bottom up, minimum distance clustering for valid affix pairs (rules). We do not assume that cluster membership is exclusive. For example, it would not make sense to determine that the null affix *-$* can belong to only one cluster. Therefore, we produce non-disjoint affix clusters.

A valid cluster of affixes is a maximal set of affixes forming pairwise valid rules: *Aff* $\subseteq$ *CAff* is a valid cluster of affixes iff

[4] where O_{11} = count($\{b_1, b_2\}$), O_{12} = count(b_2) − O_{11}, O_{21} = count(b_1) − O_{11}, $O_{22} = N - O_{11} - O_{12} - O_{21}$ and $N = \sum_{b \in \textit{CAff}}$ count(b). See table (1) for examples.

	ed	∼ed
ing	10273	21853
∼ing	27120	4119332

(a) $\chi^2 = 352678$

	le	∼le
s	122	132945
∼s	936	4044575

(b) $\chi^2 = 239.132$

	ed	∼ed
ing	2651	1310
∼ing	1490	150848

(c) $\chi^2 = 65101.6$

	le	∼le
s	20	12073
∼s	198	144008

(d) $\chi^2 = 0.631, p = 0.427$

Table 1: Affix counts in contingency tables for the valid pair *ed/ing* and spurious pair *le/s* according to *CandGen-D* in (a) and (b) and according to *CandGen-G* in (c) and (d). χ^2 test values are given under each table. Data is from NYT. Total affix token counts induced through *CandGen-D* and *CandGen-G* are N=4178578 and N=156299, respectively. A total of 2054 and 3739 affix *types* were induced for *CandGen-D* and *CandGen-G*, respectively showing that *CandGen-G* does have better coverage though it might have more noise.

1. $\forall b_1, b_2 \in$ *Aff* : $\{b_1, b_2\} \in$ *PairRule*, and
2. If $b \in$ *CAff* with $\forall b' \in$ *Aff* : $\{b, b'\} \in$ *PairRule*, then $b \in$ *Aff*.

The set of all valid affix clusters is *GroupRule*. This formulation does not rule out the existence of clusters with affixes in common.

4.4 Word clustering

We next cluster word forms into morphologically related groups. Our model assumes two word forms to be morphologically related iff (1) they occurred in the same trie *G*, (2) they have a trunk s in common that is a stem in *Stem*(G), and (3) their affixes under this stem s are members in a common valid affix cluster in *GroupRule*. Hence a single stem s can be involved in at most |*GroupRule*| conflation sets, one for each valid affix cluster. Again, the only distinction between clustering with a global trie (*Clust-G*) and clustering with several tries from the documents in a corpus (*Clust-D*) is that the former has only one trie.

We define the conflation set for a given stem $s \in$ *Stem* and valid affix cluster *Aff* $\in$ *GroupRule* as

$$\textit{Wd}(s, \textit{Aff}) = \{sb_1, sb_2 \mid b_1, b_2 \in \textit{Aff} \wedge \exists G.s \in \textit{Stem}(G) \ \wedge \ b_1, b_2 \in \textit{CAff}(s, G)\}$$

One issue that needs clarification is when the candidate generation and clustering stages use different strategies, i.e. the models *CandGen-D +Clust-G* and *CandGen-G +Clust-D*. This simply means that the *statistics*, and thus the valid *GroupRule*, are derived from either *CandGen-D* or *CandGen-G*.

4.5 Induction for languages that are both prefixal and affixal

The above approach would not fit a language that is prefixal and suffixal. Assuming we have induced separate conflation sets over a prefix trie and a suffix trie, we merge clusters between the two if they have at least one word form in common. Formally, given a set of prefix conflation sets *PCS* and a set of suffix conflation sets *SCS*, the final set of conflation sets *CS* is:

$$\textit{CS} = \{p \cup s \mid p \in \textit{PCS}, s \in \textit{SCS} \wedge \ p \cap s \neq \emptyset\}$$

5 Data

We apply our method on English and Uspanteko, an endangered Mayan language.

Learning corpora. For English, we use two subsets of the NYTimes portion in the Gigaword corpus which we will call NYT and MINI-NYT. NYT in the current study is the complete collection of articles in the New York Times from June, 2002. NYT has 10K articles, 88K types and 9M tokens. MINI-NYT is a subset of NYT with 190 articles, 15K types and 187K tokens.

The Uspanteko text, USP has 29 distinct texts, 7K types, and 50K tokens. The texts are from OKMA (Pixabaj et al., 2007) and the segmentation and labels of the interlinear glossed text annotations were checked for consistency and cleaned up (Palmer et al., 2009). All counts are for lowercased, punctuation-removed word forms.

CELEX. The CELEX lexical database (Baayen et al., 1993) has been built for Dutch, English and German and provides detailed entries that list and analyze the morphological properties of words, among other information. Using CELEX, we evaluate on types rather than tokens. The performance of the model is based on how many of the words it judges to be morphologically related overlap with the entries in CELEX. Following previous work (Schone and Jurafsky, 2000; Schone and Jurafsky,

2001; Freitag, 2005), we evaluate on inflectional clusters only, using the CELEX file listing clusters of inflectional variants. [5]

6 Experiments and evaluation

We outline our evaluation methodology, baselines, benchmarks and results, and discuss the results.

6.1 Evaluation metric

Schone and Jurafsky (2000) give definitions for correct ($\mathcal{C}$), inserted ($\mathcal{I}$), and deleted ($\mathcal{D}$) words in model-derived conflation sets in relation to a gold standard. Their formulation does not allow for multiple cluster membership of words. We extend the definition to incorporate this fact about the data. Let w be a word form. We write X_w for the clusters induced by the model that contain w, and Y_w for gold standard clusters containing w. X_w and Y_w only count words which occurred in both model and gold standard clusters. Then

$$\mathcal{C} = \sum_w \sum_{X_w} \sum_{Y_w} (|X_w \cap Y_w|/|Y_w|)$$

$$\mathcal{I} = \sum_w \sum_{X_w} \sum_{Y_w} (|X_w - (X_w \cap Y_w)|/|Y_w|)$$

$$\mathcal{D} = \sum_w \sum_{X_w} \sum_{Y_w} (|Y_w - (X_w \cap Y_w)|/|Y_w|)$$

Based on these definitions, we formulate precision (P), recall (R), and the f-score (F) as: $P = \mathcal{C}/(\mathcal{C}+\mathcal{I})$, $R = \mathcal{C}/(\mathcal{C}+\mathcal{D})$, $F = (2PR)/(P+R)$.

USP evaluation We use two different means to evaluate the performance on USP. One is the f-score derived from the above section with respect to a standard that was automatically generated from the morpheme segment tiers of the OKMA IGT. We generated the standard by taking non-hyphenated segments as the stem and clustering words with shared stems.

We also had an expert in Uspanteko manually evaluate a random subset ($N = 100$) of the model output to compensate for any failings in the standard. The evaluator determined a dominant stem for a cluster and identified words which were not related to that stem. We measured accuracy and full cluster accuracy[6] for the expert evaluations (table 4).

We experimented on Uspanteko with three different assumptions: (1) it is only prefixal; (2) it is only suffixal; (3) it is both prefixal and suffixal. We applied the assumptions of only prefixal or only suffixal to LINGUISTICA as well. The relevant results are given row headers in tables with a corresponding +P(*prefix*) or +S(suffix).

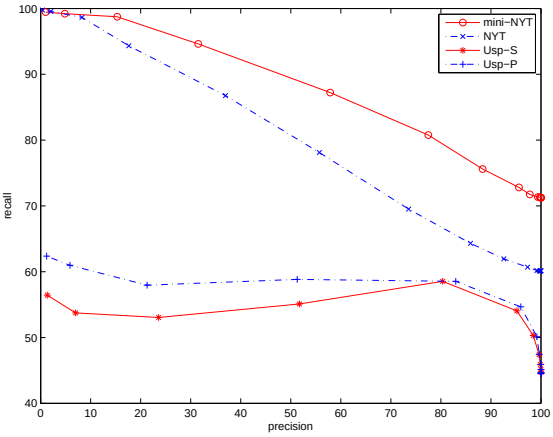


Figure 2: Precision/recall graph for baseline experiments on English, prefix USP (Usp-P) and suffix USP (Usp-S).

6.2 Baselines and benchmarks

In a *set* of baselines, we put words which share the first k characters into the same cluster. We do this for NYT, MINI-NYT, and USP in a prefix tree, and for USP in suffix tree (using the last k characters). We set the values of $0 < k < \max$, where $\max$ is the length of the longest string, and plot the results in a precision-recall graph (Fig. 2). Low k corresponds to high recall and low precision while high k shows the opposite. The contrast in morphological patterns for each language can also be seen. Because Uspanteko is morphologically complex with suffixes and prefixes, a very simple strategy cannot achieve high recall as opposed to English where it is possible to retrieve all variants with a simple prefix tree.

We use Linguistica (Goldsmith, 2001) and Morfessor (Creutz and Lagus, 2007) as benchmarks. We used the default settings for these programs. Note that comparison with these tools is not com-

[5] CELEX does have a second file listing words and their breakup into constituent morphemes for both derivation and inflection, but its use would have required additional processing that could introduce errors.

[6] Given a model cluster C_i and the "misses" for each cluster M_i, accuracy is measured as $1/N \sum_i (|C_i| - |M_i|)/(|C_i|)$ where N is the sample size. Full cluster accuracy is the number of clusters that did not have any misses over N.

	MINI-NYT			NYT		
	P	R	F	P	R	F
LINGUISTICA	64.30	**93.34**	76.15	47.50	**88.33**	61.77
MORFESSOR	45.2	87.8	59.7	63.6	69.2	66.3
CandGen-D + Clust-G	69.41	91.42	78.91	46.00	79.81	58.36
CandGen-D + Clust-D	83.47	80.36	81.89	59.02	74.50	65.86
CandGen-G + Clust-G	73.44	88.72	80.36	61.81	82.98	70.85
CandGen-G + Clust-D	**88.34**	77.95	**82.82**	**77.71**	70.24	**73.79**

Table 2: Results on English for all models in precision(P), recall(R), f-score(F) for each data set.

pletely fair. Morfessor only generates segmentations. We therefore processed Morfessor output by clustering words by assuming that the longest segment in any segmentation is the stem and evaluated this instead. Linguistica produces stems and associated suffixes so the clusters naturally follow from this output. However, Linguistica only infers either prefix or suffix patterns.

6.3 Results and discussion

The results on English are in table 2 with χ^2 test criteria of $p<0.05$ and each cell in the contingency table >5. *CandGen-G +Clust-D* had the best f-score, and easily beats the benchmarks.

This is different from our expectation that awareness of document boundaries at all stages (i.e., *CandGen-D +Clust-D*) would show the best results. The discrepancy is especially marked for the larger NYT. One important reason for this is the affix criterion itself: trunks must be longer than branches. Consider again the sample contingency tables in Table 1 that were derived from NYT through *CandGen-D* and *CandGen-G*. We had assumed at the outset that *CandGen-D* would be better able to filter out noise and would be sparser, but results show the opposite. The reason is that that short words in a global lexicon are more likely to share trunks with longer, unrelated words. This ensures that short word forms rarely generate candidate affixes. Longer words which are less likely to have spurious long branches generate the bulk of candidate suffixes and stems. This is born out by the stems that were associated with the spurious suffix pair *le/s*: *CandGen-G* has *cliente, cripp, crumb, daniel, ender, label, mccord, nag, oval, sear, stubb, whipp*. *CandGen-D* has *crumb, hand, need, sing, tab, trick, trip*. The word forms that are associated with *le/s* through the *CandGen-D* strategy are *crumble/crumbs, handle/hands, ...* Compare this with the word forms associated with the search strategy *CandGen-G* such as *clientele/clientes, cripple/crips, ...* The majority of them are not common English words; they are most probably proper names such as *LaBelle* and *Searle*. Furthermore, there is no item among the stems from the *CandGen-G* search where concatenating the stems *le* and *s* would result in both word forms being a common noun or verb as is the case with the stems from the *CandGen-D* search where all concatenated word forms are common English words. Though *CandGen-G* finds spurious stems, the counts for the spurious affix pair are suppressed (see table 1) because it is a type count rather than a token count. This results in *le/s* being properly excluded as a rule. This explains why *CandGen-D* has worse precision in general than *CandGen-G*.

The affix criterion has other minor issues. One is that it ignores the few cases where stems are shorter than affixes, such as the very common words *be, do, go*.[7] Assuming that the longest productive inflectional suffix in English is *-ing*[8], the criterion would correctly find stem candidates for *-ing* only when the stem is longer than 3 or 4 letters. Another is that the criterion, when combined with *CandGen-D*, generates candidates from *the/them/then/their/these* which cooccur frequently in documents. This is not an issue when the criterion is applied in *CandGen-G*.

Nonetheless, results show that when data sizes are small, as with USP (Table 3) and MINI-NYT, awareness of document boundaries at the candidate generation stage is beneficial to precision.

[7] The exclusion of such words in a *token* based evaluation as opposed to a *type* based evaluation would heavily penalize our approach. We are not aware, however, of any prior work in unsupervised morphology that evaluates over tokens.

[8] with occasional gemination of final consonant such as *occur* → *occurring*

	P	R	F
Ca-D + *Cl-D*	70.51	44.35	54.45
Ca-G + *Cl-G*	70.00	46.87	56.15
Ca-D + *Cl-D* + S	88.58	45.21	59.86
Ca-D + *Cl-G* + S	85.03	44.75	58.64
Ca-G + *Cl-D* + S	90.34	45.48	60.50
Ca-G + *Cl-G* + S	84.54	46.03	59.60
Ca-D + *Cl-D* + P	93.84	47.90	63.42
Ca-D + *Cl-G* + P	89.94	47.38	62.06
Ca-G + *Cl-D* + P	**95.42**	47.89	63.78
Ca-G + *Cl-G* + P	92.03	50.01	**64.80**
LINGUISTICA + S	81.14	47.60	60.00
LINGUISTICA + P	84.15	52.00	64.28
MORFESSOR	28.12	**62.28**	38.75

Table 3: Performance of models on automatically generated USP evaluation set. P: Prefix only, S: Suffix only. If there is no indication of S or P, it means model attempted to learn both

	Acc.	FAcc.	Avg. Sz.
Ca-G + *Cl-G*	98.5	79.0	2.94
LINGUISTICA	96.0	85.0	2.64
MORFESSOR	85.3	55.0	4.8

Table 4: Human expert evaluated accuracy (Acc.) and full cluster accuracy (FAcc.) of models on USP and average cluster size in words (Avg. Sz.)

However, it seems that *CandGen-G* has better coverage no matter the size of the corpus, which explains why coupling it with *Clust-D* produces overall better scores. *Clust-D* does provide a useful added constraint to mere orthographic similarity (i.e. shared trunks in a trie).

A worrisome aspect of the results is that performance degrades for large data sets (this is also true for Linguistica). However, it also hints that this method might work well for under-resourced languages. We surmise that since productive suffixes do not suffer from sparsity, even a small data set provides sufficient evidence to reach reliable conclusions about the productive morphology of some language. Increasing the size of the data merely increases the counts of spurious affixes and poses problems for a relative simple measure such as the χ^2 test. A similar result was shown in Creutz and Lagus (2005) where f-score performance of their segmentation method improved as more data was provided then decreased as the input exceeded 250K tokens in English. Their method showed continued improvement with increased data for Finnish. This hints that more data is beneficial for morphologically complex languages but not for morphologically impoverished languages.

Finally, it is also encouraging that the manual evaluation (Table 4) shows very high accuracy, as judged by a documentary linguist. Both our model and Linguistica perform very well under this evaluation.

7 Conclusion

We have presented a novel approach to unsupervised morphology acquisition that uses a very simple pipeline and does not use any thresholds other than standard ones associated with the χ^2 test. The model relies on document boundaries and correlation tests for filtering spurious stems and affixes. The model compares favorably to Linguistica and Morfessor, two models that employ much more complex strategies and rely on fine-tuned parameters. We found that the use of document boundaries is especially beneficial with small datasets, which is promising for the application of this model to under-resourced languages. For large datasets, global candidate generation outperformed document-aware candidate generation at the task of filtering out spurious stems, but document-aware clustering does improve precision and overall performance.

In this paper we have addressed one aspect of morphology acquisition, segmentation and clustering. Extending the approach is straightforward, for example, substituting more sophisticated data structures or statistical tests for the current ones. In particular, we will move from the use of document boundaries to a flexible notion of textual distance to estimate likelihood of morphological relatedness.

Acknowledgments

This work is funded by NSF grant BCS 06651988 "Reducing Annotation Effort in the Documentation of Languages using Machine Learning and Active Learning." Thanks to Alexis Palmer, Telma Kaan Pixabaj, Elias Ponvert, and the anonymous reviewers.

References

R. H. Baayen, R. Piepenbrock, and H. van Rijn. 1993. *The CELEX lexical database on CD-ROM*. Linguistic Data Consortium, Philadelphia, PA.

M. Baroni, J. Matiasek, and H. Trost. 2002. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *ACL '02 workshop on Morphological and phonological learning*, pages 48–57.

D. Bernhard. 2005. Unsupervised morphological segmentation based on segment predictability and word segments alignment. In *Proceedings of Morpho Challenge 2005*, pages 18–27.

S. Bordag. 2005. Two-step approach to unsupervised morpheme segmentation. In *Proceedings of Morpho Challenge 2005*, pages 23–27.

E. Chan. 2006. Learning Probabilistic Paradigms for Morphology in a Latent Class Model. In *ACL SIGPHON '06*, pages 69–78.

M. Creutz and K. Lagus. 2002. Unsupervised discovery of morphemes. In *ACL '02 workshop on Morphological and phonological learning-Volume 6*, pages 21–30.

M. Creutz and K. Lagus. 2004. Induction of a simple morphology for highly-inflecting languages. In *ACL SIGPHON '04*, pages 43–51.

M. Creutz and K. Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In *AKRR '05*, pages 106–113.

M. Creutz and K. Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1):3.

S. Dasgupta and V. Ng. 2007. High-performance, language-independent morphological segmentation. In *NAACL-HLT*, pages 155–163.

V. Demberg. 2007. A language-independent unsupervised model for morphological segmentation. In *ACL '07*, volume 45, page 920.

D. Freitag. 2005. Morphology induction from term clusters. In *CoNLL '05*.

E. Gaussier. 1999. Unsupervised learning of derivational morphology from inflectional lexicons. In *ACL workshop on Unsupervised Methods in Natural Language Learning*.

J. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Comp. Ling.*, 27(2):153–198.

J. Goldsmith. 2006. An algorithm for the unsupervised learning of morphology. *Natural Language Engineering*, 12(04):353–371.

S.J. Goldwater. 2007. *Nonparametric Bayesian models of lexical acquisition*. Ph.D. thesis, Brown University.

M.A. Hafer and S.F. Weiss. 1974. Word Segmentation by Letter Successor Varieties. *Information Storage and Retrieval*, 10:371–385.

H. Hammarström. 2006. A naive theory of affixation and an algorithm for extraction. In *ACL SIGPHON '06*, pages 79–88, June.

S. Keshava and E. Pitler. 2005. A simpler, intuitive approach to morpheme induction. In *Proceedings of Morpho Challenge 2005*, pages 28–32.

C. Monson. 2004. A framework for unsupervised natural language morphology induction. In *Proceedings of the Student Workshop at ACL*, volume 4.

Alexis Palmer, Taesun Moon, and Jason Baldridge. 2009. Evaluating automation strategies in language documentation. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 36–44, Boulder, CO.

T.C. Pixabaj, M.A. Vicente Méndez, M. Vicente Méndez, and O.A. Damián. 2007. Text collections in Four Mayan Languages. Archived in The Archive of the Indigenous Languages of Latin America.

P. Schone and D. Jurafsky. 2000. Knowledge-free induction of morphology using latent sematic analysis. In *CoNLL-2000 and LLL-2000*.

P. Schone and D. Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *NAACL '01*, pages 1–9.

M.G. Snover, G.E. Jarosz, and M.R. Brent. 2002. Unsupervised learning of morphology using a novel directed search algorithm: taking the first step. In *ACL '02 workshop on Morphological and phonological learning*, pages 11–20.

B. Snyder and R. Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *ACL '08*.

D. Yarowsky and R. Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *ACL '00*, pages 207–216.

D. Yarowsky, G. Ngai, and R. Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *HLT '01*.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL '95*, pages 189–196.

The infinite HMM for unsupervised PoS tagging

Jurgen Van Gael
Department of Engineering
University of Cambridge
jv249@cam.ac.uk

Andreas Vlachos
Computer Laboratory
University of Cambridge
av308@cl.cam.ac.uk

Zoubin Ghahramani
Department of Engineering
University of Cambridge
zoubin@eng.cam.ac.uk

Abstract

We extend previous work on fully unsupervised part-of-speech tagging. Using a non-parametric version of the HMM, called the infinite HMM (iHMM), we address the problem of choosing the number of hidden states in unsupervised Markov models for PoS tagging. We experiment with two non-parametric priors, the Dirichlet and Pitman-Yor processes, on the Wall Street Journal dataset using a parallelized implementation of an iHMM inference algorithm. We evaluate the results with a variety of clustering evaluation metrics and achieve equivalent or better performances than previously reported. Building on this promising result we evaluate the output of the unsupervised PoS tagger as a direct replacement for the output of a fully supervised PoS tagger for the task of shallow parsing and compare the two evaluations.

1 Introduction

Many Natural Language Processing (NLP) tasks are commonly tackled using supervised learning approaches. These learning methods rely on the availability of labeled datasets which are usually produced by expensive manual annotation. For some tasks, we have the choice to use unsupervised learning approaches. While they do not necessarily achieve the same level of performance, they are appealing as unlabeled data is usually abundant. In particular, for the purpose of exploring new domains and languages, obtainining labeled material can be prohibitively expensive and unsupervised learning methods are a very attractive choice. Recent work (Johnson, 2007; Goldwater and Griffiths, 2007; Gao and Johnson, 2008) explored the task of part-of-speech tagging (PoS) using unsupervised Hidden Markov Models (HMMs) with encouraging results. PoS tagging is a standard component in many linguistic processing pipelines, so any improvement on its performance is likely to impact a wide range of tasks.

It is important to point out that a completely unsupervised learning method will discover the statistics of a dataset *according to a particular model choice* but these statistics might not correspond exactly to our intuition about PoS tags. Johnson (2007) and Gao & Johnson (2008) assume that words are generated by a hidden Markov model and find that the resulting states strongly correlate with POS tags. Nonetheless, *identifying* the HMM states with appropriate POS tags is hard. Because many evaluation methods often require POS tags (rather than HMM states) this identification problem makes unsupervised systems difficult to evaluate.

One potential solution is to add a small amount of supervision as in Goldwater & Griffiths (2007) who assume a dictionary of frequent words associated with possible PoS tags extracted from a labeled corpus. Although this technique improves performance, in this paper we explore the completely unsupervised approach. The reason for this is that better unsupervised approaches provide us with better starting points from which to explore how and where to incorporate supervision.

In previous work on unsupervised PoS tagging a main question was how to set the number of hidden states appropriately. Johnson (2007) reports results for different numbers of hidden states but it is unclear how to make this choice a priori, while Goldwater & Griffiths (2007) leave this question as future work.

It is not uncommon in statistical machine learning to distinguish between parameters of a model and the capacity of a model. E.g. in a clustering context, the choice for the number of clusters (capacity) and the parameters of each cluster are often

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 678–687,
Singapore, 6-7 August 2009.

treated differently: the latter are estimated using algorithms like EM, MCMC or Variational Bayes while the former is chosen using common sense, heuristics or in a Bayesian framework maybe using evidence maximization.

Non-parametric Bayesian methods are a class of probability distributions which explicitly treat the capacity of a model as "just another parameter". Potential advantages are

- the model capacity can automatically adjust to the amount of data: e.g. when clustering a very small dataset, it is unlikely that many fine grained clusters can be distinguished,

- inference can be more efficient: e.g. instead of running full inference for different model capacities and then choosing the best capacity (according to some choice of "best"), inference in non-parametric Bayesian methods integrates the capacity search in one algorithm. This is particularly advantageous when parameters other than capacity need to be explored, since it reduces signifcantly the number of experiments needed.

None of these potential advantages are guaranteed and in this paper we investigate these two aspects for the task of unsupervised PoS tagging.

The contributions in this paper extend previous work on unsupervised PoS tagging in five ways. First, we introduce the use of a non-parametric version of the HMM, namely the infinite HMM (iHMM) (Beal et al., 2002) for unsupervised PoS tagging. This answers an open problem from Goldwater & Griffiths (2007). Second, we carefully implemented a parallelized version of the inference algorithms for the iHMM so we could use it on the Wall Street Journal Penn Treebank dataset. Third, we introduce a new variant of the iHMM that builds on the Pitman-Yor process. Fourth, we evaluate the results with a variety of clustering evaluation methods and achieve equivalent or better performances than previously reported. Finally, building on this promising result we use the output of the unsupervised PoS tagger as a direct replacement for the output of a fully supervised PoS tagger for the task of shallow parsing. This evaluation enables us to assess the applicability of an unsupervised PoS tagging method and provides us with means of comparing its performance against a supervised PoS tagger.

The rest of the paper is structured as follows: in section 2 we introduce the iHMM as a non-parametric version of the Bayesian HMM used in previous work on unsupervised PoS tagging. Then, in section 3 we describe some details of our implementation of the iHMM. In section 4 we present a variety of evaluation metrics to compare our results with previous work. Finally, in section 5 we report our experimental results. We conclude this paper with a discussion of ongoing work and experiments.

2 The Infinite HMM

In this section, we describe a non-parametric hidden Markov model known as the infinite HMM (iHMM) (Beal et al., 2002; Teh et al., 2006). As we show below, this model is flexible in the number of hidden states which it can accomodate. In other words, *the capacity is an uncertain quantity* with an a priori infinite range that is a posteriori inferred by the data. It is instructive to first review the finite HMM and its Bayesian treatment: for one, it is the model that has been used in previous work on unsupervised PoS tagging, secondly it allows us to better understand the iHMM.

The Bayesian HMM A finite first-order HMM consists of a hidden state sequence $\boldsymbol{s} = (s_1, s_2, \ldots, s_T)$ and a corresponding observation sequence $\boldsymbol{y} = (y_1, y_2, \ldots, y_T)$. Each state variable s_t can take on a finite number of states, say $1 \ldots K$. Transitions between states are governed by Markov dynamics parameterized by the transition matrix $\boldsymbol{\pi}$, where $\pi_{ij} = p(s_t = j | s_{t-1} = i)$, while the initial state probabilities are $\pi_{0i} = p(s_1 = i)$. For each state $s_t \in \{1 \ldots K\}$ there is a parameter ϕ_{s_t} which parameterizes the observation likelihood for that state: $y_t | s_t \sim F(\phi_{s_t})$. Given the parameters $\{\pi_0, \boldsymbol{\pi}, \boldsymbol{\phi}, K\}$ of the HMM, the joint distribution over hidden states $\boldsymbol{s}$ and observations $\boldsymbol{y}$ can be written (with $s_0 = 0$):

$$p(\boldsymbol{s}, \boldsymbol{y} | \pi_0, \boldsymbol{\pi}, \boldsymbol{\phi}, K) = \prod_{t=1}^{T} p(s_t | s_{t-1}) p(y_t | s_t)$$

As Johnson (2007) clearly explained, training the HMM with EM leads to poor results in PoS tagging. However, we can easily treat the HMM in a fully Bayesian way (MacKay, 1997) by introducing priors on the parameters of the HMM. With no further prior knowledge, a typical prior for the transition (and initial) probabilities are symmetric Dirichlet distributions. This corresponds to our

belief that, a priori, each state is equally likely to transition to every other state. Also, it is commonly known that the parameter of a Dirichlet distribution controls how sparse its samples are. In other words, by making the hyperprior on the Dirichlet distribution for the rows of the transition matrix small, we can encode our belief that any state (corresponding to a PoS tag in this application context) will only be followed by a small number of other states. As we explain below, we will be able to include this desirable property in the non-parametric model as well. Secondly, we need to introduce a prior on the observation parameters ϕ_k. Without any further prior knowledge, a convenient choice here is another symmetric Dirichlet distribution with sparsity inducing hyperprior. This encodes our belief that only a subset of the words correspond to a particular state.

The Infinite HMM A first naïve way to obtain a non-parametric HMM with an infinite number of states might be to use symmetric Dirichlet priors over the transition probabilities with parameter α/K and take $K \to \infty$. This approach unfortunately does not work: $\alpha/K \to 0$ when $K \to \infty$ and hence the rows of the matrix will become "infinitely sparse". Since the sum of the entries must sum to one, the rows of the transition matrix will be zero everywhere and all its mass in a random location. Unfortunately, this random location is out of an infinite number of possible locations and hence with probability 1 will be different for all the rows. As a consequence, at each timestep the HMM moves to a new state and will never revisit old states. As we shall see shortly, we can fix this by using a hierarchical Bayesian formalism where the Dirichlet priors on the rows have a shared parameter.

Before moving on to the iHMM, let us look at the finite HMM from a different perspective. The finite HMM of length T with K hidden states can be seen as a sequence of T finite mixture models. The following equation illustrates this idea: conditioned on the previous state s_{t-1}, the marginal probability of observation y_t can be written as:

$$\begin{aligned} p(y_t|s_{t-1}=k) &= \sum_{s_t=1}^{K} p(s_t|s_{t-1}=k)p(y_t|s_t), \\ &= \sum_{s_t=1}^{K} \pi_{k,s_t} p(y_t|\phi_{s_t}). \end{aligned} \quad (1)$$

The variable $s_{t-1} = k$ specifies the mixing weights $\pi_{k,\cdot}$ for the mixture distribution, while s_t indexes the mixture component generating the observation y_t. In other words, equation (1) says that each row of the transition matrix $\boldsymbol{\pi}$ specifies a different mixture distribution over the same set of K mixture components $\boldsymbol{\phi}$.

Our second attempt to define a non-parametric version of the hidden Markov model is to replace the finite mixture by an infinite mixture. The theory of Dirichlet process mixtures (Antoniak, 1974) tells us exactly how to do this. A draw $G \sim DP(\alpha, H)$ from a Dirichlet process (DP) with base measure H and concentration parameter $\alpha \geq 0$ is a discrete distribution which can be written as an infinite mixture of atoms

$$G(\cdot) = \sum_{i=1}^{\infty} \pi_i \delta_{\phi_i}(\cdot)$$

where the ϕ_i are i.i.d. draws from the base measure H, $\delta_{\phi_i}(\cdot)$ represents a point distribution at ϕ_i and $\pi_i = v_i \prod_{l=1}^{i-1}(1 - v_l)$ where each $v_l \sim \mathsf{Beta}(1, \alpha)$. The distribution over π_i is called a stick breaking construction and is essentially an infinite dimensional version of the Dirichlet distribution. We refer to Teh et al. (2006) for more details.

Switching back to the iHMM our next step is to introduce a DP G_j for each state $j \in \{1 \cdots \infty\}$; we write $G_j(\cdot) = \sum_{i=1}^{\infty} \pi_i^j \delta_{\phi_i^j}(\cdot)$. There is now a parameter for each state j and each index $i \in \{1, 2, \cdots, \infty\}$. Next, we draw the datapoint at timestep t given that the previous datapoint was in state s_{t-1} by drawing from DP $G_{s_{t-1}}$. We first select a mixture component s_t from the vector $\pi_{s_{t-1},\cdot}$ and then sample a datapoint $y_t \sim F(\phi_{s_{t-1},s_t})$ so we get the following distribution for y_t

$$p(y_t|\alpha, s_{t-1}) = \sum_{s_t=1}^{\infty} \pi_{s_{t-1},s_t} p(y_t|\phi_{s_{t-1},s_t}).$$

This is almost the non-parametric equivalent of equation (1) but there is a subtle difference: each G_j selects their own set of parameters $\phi_{\cdot}^j$. This is unfortunate as it means that the output distribution would not be the same for each state, it would depend on which state we were moving to! Luckily, we can easily fix this: by introducing an intermediate distribution $G_0 \sim DP(\gamma, H)$ and let $G_j \sim DP(\alpha, G_0)$ we enforce that the i.i.d. draws $\phi_{\cdot}^j$ are draws from a discrete distribution (since G_0

is a draw from a Dirichlet process) and hence all G_j will share the same infinite set of atoms as chosen by G_0. Figure 1 illustrates the graphical model for the iHMM.

The iHMM with Pitman-Yor Prior The Dirichlet process described above defines a very specific distribution over the number of states in the iHMM. One particular generalization of the Dirichlet process that has been studied in the NLP literature before is the *Pitman-Yor* process. Goldwater et al. (2006) have shown that the Pitman-Yor distribution can more accurately capture power-law like distributions that frequently occur in natural language.

More specifically, a draw $G \sim PY(d, \alpha, H)$ from a Pitman-Yor process (PY) with base measure H, discount parameter $0 \leq d < 1$ and concentration parameter $\alpha > -d$ is a discrete distribution which can be written as an infinite mixture of atoms

$$G(\cdot) = \sum_{i=1}^{\infty} \pi_i \delta_{\phi_i}(\cdot)$$

where the ϕ_i are i.i.d. draws from the base measure H, $\delta_{\phi_i}(\cdot)$ represents a point distribution at ϕ_i and $\pi_i = v_i \prod_{l=1}^{i-1}(1 - v_l)$ where each $v_l \sim \mathsf{Beta}(1-d, \alpha+ld)$. Note the similarity to the DP: in fact, the DP is a special case of PY with $d = 0$.

In our experiments, we constructed an iHMM where the $DP(\alpha, H)$ base measure G_0 is replaced with its two parameter generalization $PY(d, \alpha, H)$. Because the Dirichlet and Pitman-Yor processes only differ in the way π is constructed, without loss of generality we will describe hyper-parameter choice and inference in the context of the iHMM with Dirichlet process base measure.

Hyperparameter Choice The description above shows that there are 4 parameters which we must specify: the base measure H, the output distribution $p(y_t|\phi_{s_t})$, the discount[1] and concentration[2] parameters d, γ for G_0 and the concentration parameter α for the DP's G_j. Just as in the finite case, the base measure H is the prior distribution on the parameter ϕ of $p(y_t|\phi_{s_t})$. We chose to use a symmetric Dirichlet distribution with parameter δ over the word types in our corpus. Since we do not know the sparsity level δ of the output distributions we decided to learn this parameter from the data. We initially set a vague Gamma prior over δ but soon realized that as we expect hidden states in the iHMM to correspond to PoS tags, it is unrealistic to expect each state to have the same sparsity level. Hence we chose a Dirichlet process as the prior for δ; this way we end up with a small discrete set of sparsity levels: e.g. we can learn that states corresponding to verbs and nouns share one sparsity level while states correpsonding to determiners have their own (much sparser) sparsity level. For the output distribution $p(y_t|\phi_{s_t})$ we chose a simple multinomial distribution.

The hyperparameters d and γ mostly control the number of states in the iHMM while - as we discussed above - α controls the sparsity of the transition matrix. In the experiments below we report both fixing the two parameters and learning them by sampling (using vague Gamma hyperpriors). Because of computational constraints, we chose to use vague Bayesian priors for all hyperparameters rather than run the whole experiment over a grid of "reasonable" parameter settings and use the best ones according to cross validation.

3 Inference

The Wall Street Journal part of the Penn Treebank that was used for our experiments contains about one million words. In the non-parametric Bayesian literature not many algorithms have been described that scale into this regime. In this section we describe our parallel implementation of the iHMM which can easily handle a dataset of this scale.

There is a wealth of evidence (Scott, 2002; Gao and Johnson, 2008) in the machine learning literature that Gibbs sampling for Markov models leads to slow mixing times. Hence we decided our starting point for inference needs to be based on dynamic programming. Because we didn't have a good idea for the number of states that we were going to end up with, we prefered the beam sampler of Van Gael et al. (2008) over a finite truncation of the iHMM. Moreover, the beam sampler also introduces a certain amount of sparsity in the dynamic program which can speed up computations (potentially at the cost of slower mixing).

The beam sampler is a blocked Gibbs sampler where we alternate between sampling the parameters (transition matrix, output parameters), the state sequence and the hyperparameters. Sam-

[1] for Pitman-Yor base measure

[2] for both Dirichlet and Pitman-Yor base measures

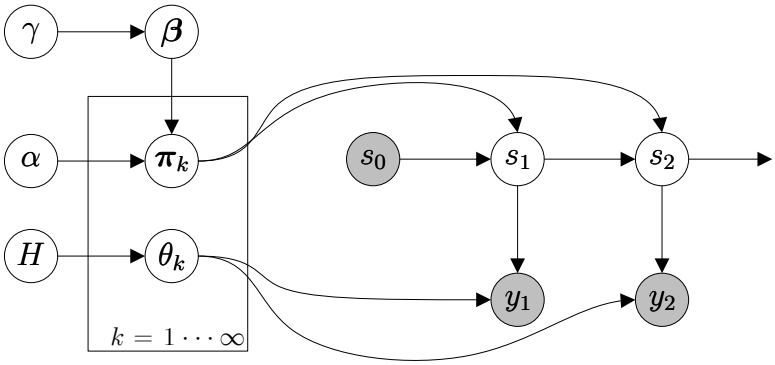


Figure 1: The graphical model for the iHMM. The variable β represents the mixture for the DP G_0.

pling the transition matrix and output distribution parameters requires computing their sufficient statistics and sampling from a Dirichlet distribution; we refer to the beam sampling paper for details. For the hyperparameters we use standard Gibbs sampling. We briefly sketch the resampling step for the state sequence for a single sequence of data (sentence of words). Running standard dynamic programming is prohibitive because the state space of the iHMM is infinitely large. The central idea of the beam sampler is to adaptively truncate the state space of the iHMM and run dynamic programming. In order to truncate the state space, we sample an auxilary variable u_t for each word in the sequence from the distribution $u_t \sim \mathsf{Uniform}(0, \pi_{s_{t-1}s_t})$ where π represents the transition matrix.

Intuitively, when we sample $u_{1:T}|s_{1:T}$ according to the distribution above, the only valid samples are those for which the u_t are smaller than the transition probabilities of the state sequence $s_{1:T}$. This means that when we sample $s_{1:T}|u_{1:T}$ at a later point, it must be the case that the u_t's are still smaller than the new transition probabilities. This significantly reduces the set of valid state sequences that we need to consider. More specifically, Van Gael et al. (2008) show that we can compute $p(s_t|y_{1:t}, u_{1:t})$ using the following dynamic programming recursion $p(s_t|y_{1:t}, u_{1:t}) =$

$$p(y_t|s_t) \sum_{s_{t-1}:u_t<\pi_{s_{t-1},s_t}} p(s_{t-1}|y_{1:t-1}, u_{1:t-1}).$$

The summation $\sum_{s_{t-1}:u_t<\pi_{s_{t-1},s_t}}$ ensures that this computation remains finite. When we compute $p(s_t|y_{1:t}, u_{1:t})$ for $t \in \{1 \cdots T\}$, we can easily sample s_T and using Bayes rule backtrack sample every other s_t. It can be shown that this procedure produces samples from the exact posterior.

Notice that the dynamic program only needs to perform computation when $u_t < \pi_{s_{t-1},s_t}$. A careful implementation of the beam sampler consists of preprocessing the transition matrix π and sorting its elements in descending order. We can then iterate over the elements of the transition matrix starting from the largest element and stop once we reach the first element of the transition matrix smaller than u_t. In our experiments we found that this optimization reduces the amount of computation per sentence by an order of magnitutde.

A second optimization which we introduced is to use the map-reduce paradigm (Dean and Ghemawat, 2004) to parallelize our computations. More specifically, after we preprocess the transition matrix, the dynamic program computations are independent for each sentence in the dataset. This means we can perform each dynamic program in parallel; in other words our "map" consists of running the dynamic program on one sentence in the dataset. Next, we need to resample the transition matrix and output distribution parameters. In order to do so we need to compute their sufficient statistics: the number of transitions from state to state and the number of emissions of each word out of each state. Our "reduce" function consists of computing the sufficient statistics for each sentence and then aggregating the statistics for the whole dataset. Our implementation runs on a quad-core shared memory architecture and we find an almost linear speedup going from one to four cores.

4 Evaluation

Evaluating unsupervised PoS tagging is rather difficult mainly due to the fact that the output of such

systems are not actual PoS tags but state identifiers. Therefore it is impossible to evaluate performance against a manually annotated gold standard using accuracy. Recent work (Goldwater and Griffiths, 2007; Johnson, 2007; Gao and Johnson, 2008) on this task explored a variety of methodologies to address this issue.

The most common approach followed in previous work is to evaluate unsupervised PoS tagging as clustering against a gold standard using the Variation of Information (VI) (Meilă, 2007). VI assesses homogeneity and completeness using the quantities $H(C|K)$ (the conditional entropy of the class distribution in the gold standard given the clustering) and $H(K|C)$ (the conditional entropy of clustering given the class distribution in the gold standard). However, as Gao & Johnson (2008) point out, VI is biased towards clusterings with a small number of clusters. A different evaluation measure that uses the same quantities but weighs them differently is the V-measure (Rosenberg and Hirschberg, 2007), which is defined in Equation 2 by setting the parameter β to 1.

$$h = 1 - \frac{H(C|K)}{H(C)}$$
$$c = 1 - \frac{H(K|C)}{H(K)}$$
$$V_\beta = \frac{(1+\beta)hc}{(\beta h) + c} \quad (2)$$

Vlachos et al. (2009) noted that V-measure favors clusterings with a large number of clusters. Both of these biases become crucial in our experiments, since the number of clusters (states of the iHMM) is not fixed in advance. Vlachos et al. proposed a variation of the V-measure, V-beta, that adjusts the balance between homogeneity and completeness using the parameter β in Eq. 2.

It is worth mentioning that, unlike V-measure and V-beta, VI scores are not normalized and therefore they are difficult to interpret. Meilă (2007) presented two normalizations, acknowledging the potential disadvantages they have. The first one normalizes VI by $2\log(max(|K|,|C|))$, which is inappropriate when the number of clusters discovered $|K|$ changes between experiments. The second normalization involves the quantity $\log N$ which is appropriate when comparing different algorithms on the same dataset (N is the number of instances). However, this quantity depends exclusively on the size of the dataset and hence if the dataset is very large it can result in normalized VI scores misleadingly close to 100%. This does not affect rankings, i.e. a better VI score will also be translated into a better normalized VI score. In our experiments, we report results only with the un-normalized VI scores, V-measure and V-beta.

All the evaluation measures mentioned so far evaluate PoS tagging as a clustering task against a manually annotated gold standard. While this is reasonable, it still does not provide means of assessing the performance in a way that would allow comparisons with supervised methods that output actual PoS tags. Even for the normalized measures V-measure and V-beta, it is unclear how their values relate to accuracy levels. Gao & Johnson (2008) partially addressed this issue by mapping states to PoS tags following two different strategies, cross-validation accuracy, and greedy 1-to-1 mapping, which both have shortcomings. We argue that since an unsupervised PoS tagger is trained without taking any gold standard into account, it is not appropriate to evaluate against a particular gold standard, or at least this should not be the sole criterion. The fact that different authors use different versions of the same gold standard to evaluate similar experiments (e.g. Goldwater & Griffiths (2007) versus Johnson (2007)) supports this claim. Furthermore, PoS tagging is seldomly a goal in itself, but it is a component in a linguistic pipeline.

In order to address these issues, we perform an extrinsic evaluation using a well-explored task that involves PoS tags. While PoS tagging is considered a pre-processing step in many natural language processing pipelines, the choice of task is restricted by the lack of real PoS tags in the output of our system. For our purposes we need a task that relies on discriminating between PoS tags rather than the PoS tag semantics themselves, in other words, a task in which knowing whether a word is tagged as noun instead of a verb is equivalent to knowing it is tagged as state 1 instead of state 2. Taking these considerations into account, in Section 5 we experiment with shallow parsing in the context of the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000) in which very good performances were achieved using only the words with their PoS tags. Our intuition is that if the iHMM (or any unsupervised PoS tagging

method) has a reasonable level of performance, it should improve on the performance of a system that does not use PoS tags. Moreover, if the performance is very good indeed, it should get close to the performance of a system that uses real PoS tags, provided either by human annotation or by a good supervised system. Similar extrinsic evaluation was performed by Biemann et al. (2007). It is of interest to compare the results between the clustering evaluation and the extrinsic one.

A different approach in evaluating non-parametric Bayesian models for NLP is state-splitting (Finkel et al., 2007; Liang et al., 2007). In this setting, the model is used in order to refine existing annotation of the dataset. While this approach can provide us with some insights and interpretable results, the use of existing annotation influences the output of the model. In this work, we want to verify whether the output of the iHMM (without any supervision) can be used instead of that of a supervised system.

5 Experiments

In all our experiments, the Wall Street Journal (WSJ) part of the Penn Treebank was used. As explained in Section 4, we evaluate the output of the iHMM in two ways, as clustering with respect to a gold standard and as direct replacement of the PoS tags in the task of shallow parsing. In each experiment, we obtain a sample from the iHMM over all the sections of WSJ. The states for sections 15-18 and 20 of the WSJ (training and testing sets respectvely in the CoNLL shared task) are used for the evaluation based on shallow parsing, while the remaining sections are used for evaluation against the WSJ gold standard PoS tags using clustering evaluation measures.

As described in Section 2 we performed three runs with the iHMM: one run with DP prior and fixed γ, α, one with PY prior and fixed d, γ, α and one with DP prior but where we learn the hyperparameters γ, α from the data. Our inference algorithm uses 1000 burn-in iterations after which we collect a sample every 1000 iterations. Our inference procedure is annealed during the first 1000 burnin and 2400 iterations by powering the likelihood of the output distribution with a number that smoothly increases from 0.4 to 1.0 over the 3400 first iterations. The numbers of iterations reported in the remainder of the section refer to the iterations after burn-in. We initialized the sampler by: a) sampling the hyperparameters from the prior where applicable, b) uniformly assign each word one out of 20 iHMM states. For the DP run with fixed parameters, we chose $\alpha = 0.8$ to encourage some sparsity in the transition matrix and $\gamma = 5.0$ to allow for enough hidden states. For the PY run with fixed parameters, we chose $\alpha = 0.8$ for similar reasons and $d = 0.1$ and $\gamma = 1.0$. We point out that one weakness of MCMC methods is that they are hard to test for convergence. We chose to run the simulations until they became prohibitively expensive to obtain a new sample.

First, we present results using clustering evaluation measures which appear in the figures of Table 1. The three runs exhibit different behavior. The number of states reached by the iHMM with fixed parameters using the DP prior stabilizes close to 50 states, while for the experiment with learnt hyperparameters the number of states grows more rapidly, reaching 194 states after 8,000 iterations. With the PY prior, the number of states reached grows less rapidly reaching 90 states. All runs achieve better performances with respect to all the measures used as the number of iterations grows. An exception is that VI scores tend to increase (lower VI scores are better) when the number of states grows larger than the gold standard. It is interesting to notice how the measures exhibit different biases, in particular that VI penalizes the larger numbers of states discovered in the DP run with learnt parameters as well as the run with the PY prior, compared to the more lenient scores provided by V-measure and V-beta. The latter though assigns lower scores to the DP run with learnt parameters because it takes into account that the high homogeneity is achieved using even more states. Finally, the interpretability of these scores presents some interest. For example, in the run with fixed parameters using the DP prior, after burn-in VI was 4.6, which corresponds to 76.65% normalized VI score, while V-measure and V-beta were 12.7% and 9% respectively. In 8,000 iterations after burn-in, VI was 3.94 (80.3% when normalized), while V-measure and V-beta were 53.3%, since the number of states was almost the same as the number of unique PoS tags in the gold standard.

The closest experiment to ours is the one by Gao & Johnson (2008) who run their Bayesian HMM over the whole WSJ and evaluated against the full gold standard, the only difference being is that we exclude the CoNLL shared task sec-

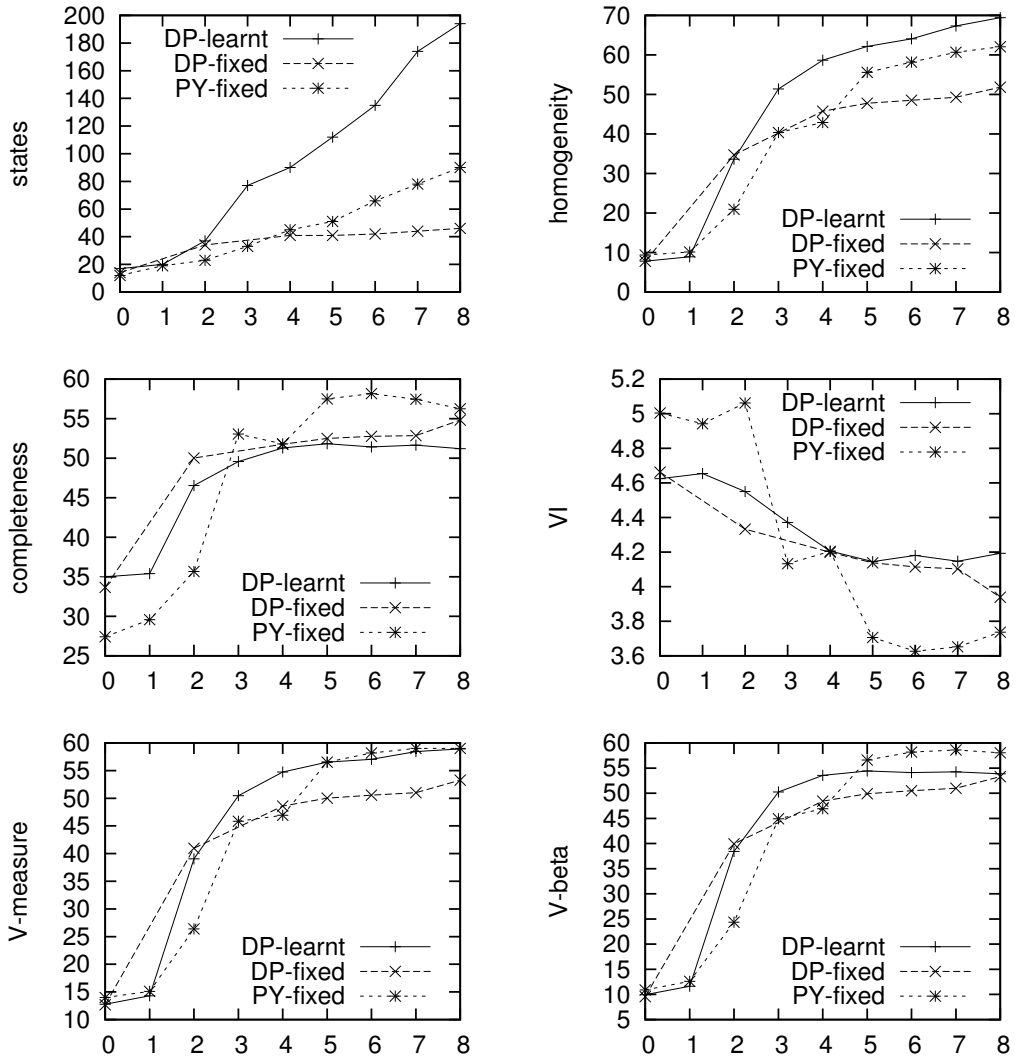


Table 1: Performance of the three iHMM runs according to clustering evaluation measures against number of iteretions (in thousands).

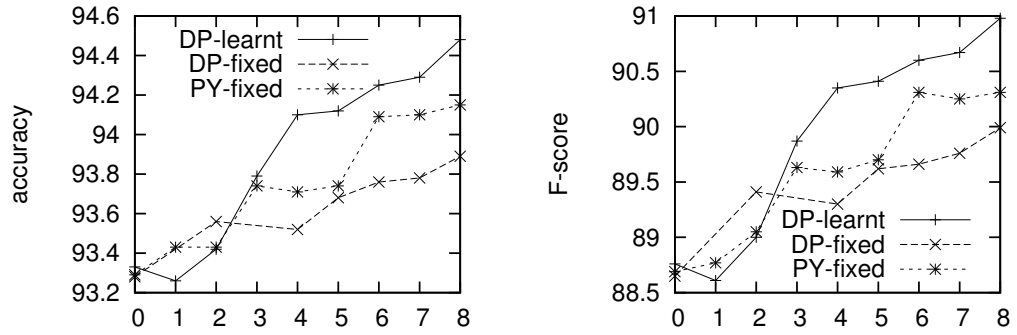


Table 2: Performance of the output of the three iHMM runs when used in shallow parsing against number of iteretions (in thousands).

tions from our evaluation, which leaves us with 19 sections instead of 24. Their best VI score was 4.03886 which they achieved using the collapsed, sentence-blocked Gibbs sampler with the number of states fixed to 50. The VI score achieved by the iHMM with fixed parameters using the PY prior reaches 3.73, while using the DP prior VI reaches 4.32 with learnt parameters and 3.93 with fixed

parameters. These results, even if they are not directly comparable, are on par with the state-of-the-art, which encouraged us to proceed with the extrinsic evaluation.

For the experiments with shallow parsing we used the CRF++ toolkit[3] which has an efficient implementation of the model introduced by Sha & Pereira (2003) for this task. First we ran an experiment using the words and the PoS tags provided in the shared task data and the performances obtained were 96.07% accuracy and 93.81% F-measure. The PoS tags were produced using the Brill tagger (Brill, 1994) which employs tranformation-based learning and was trained using the WSJ corpus. Then we ran an experiment removing the PoS tags altogether, and the performances were 93.25% accuracy and 88.58% F-measure respectively. This gave us some indication as to what the contribution of the PoS tags is in the context of the shallow parsing task at hand.

The experiments using the output of the iHMM as PoS tags for shallow parsing are presented in Table 2. The best performance achieved was 94.48% and 90.98% in accuracy and F-measure, which is 1.23% and 2.4% better respectively than just using words, but worse by 1.57% and 2.83% compared to using the supervised PoS tagger output. Given that the latter is trained on WSJ we believe that this is a good result. Interestingly, this was obtained by using the last sample from the iHMM run using the DP prior with learnt parameters which has worse overall clustering evaluation scores, especially in terms of VI. This sample though has the best homogeneity score (69.39%). We believe that homogeneity is more important than the overall clustering score due to the fact that, in the application considered, it is probably worse to assign tokens that belong to different PoS tags to the same state, e.g. verb and adverbs, rather than generate more than one state for the same PoS. This is likely to be the case in tasks where we are interested in distinguishing between PoS tags rather than the actual tag itself. Also, clustering evaluation measures tend to score leniently consistent mixing of members of different classes in the same cluster. However, such mixing results in consistent noise when the clustering output becomes input to a machine learning method, which is harder to deal with.

[3] http://crfpp.sourceforge.net/

6 Conclusions - Future Work

In the context of shallow parsing we saw that the performance of the iHMM does not match the performance of a supervised PoS tagger but does lead to a performance increase over a model using only words as features. Given that it was constructed without any need for human annotation, we believe this is a good result. At the same time though, it suggests that it is still some way from being a direct drop-in replacement for a supervised method. We argue that the extrinsic evaluation of unsupervised PoS tagging performed in this paper is quite informative as it allowed us to assess our results in a more realistic context. In this work we used shallow parsing for this, but we are considering other tasks in which we hope that PoS tagging performance will be more crucial.

Our experiments also suggest that the number of states in a Bayesian non-parametric model can be rather unpredictable. On one hand, this is a strong warning towards inference algorithms which perform finite truncation of non-parametric models. On the other hand, the remarkable difference in behavior between the DP with fixed and learned priors suggests that more research is needed towards understanding the influence of hyperparameters in Bayesian non-parametric models.

We are currently experimenting with a semi-supervised PoS tagger where we let the transition matrix of the iHMM depend on annotated PoS tags. This model allows us to: a) use annotations whenever they are available and do unsupervised learning otherwise; b) use the power of non-parametric methods to possibly learn more fine grained statistical structure than tag sets created manually.

On the implementation side, it would be interesting to see how our methods scale in a distributed map-reduce architecture where network communication overhead becomes an issue.

Finally, the ultimate goal of our investigation is to do unsupervised PoS tagging using web-scale datasets. Although the WSJ corpus is reasonably sized, our computational methods do not currently scale to problems with one or two order of magnitude more data. We will need new breakthroughs to unleash the full potential of unsupervised learning for NLP.

References

Charles E. Antoniak. 1974. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174.

M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. 2002. The infinite hidden markov model. *Advances in Neural Information Processing Systems*, 14:577 – 584.

Chris Biemann, Claudio Giuliano, and Alfio Gliozzo. 2007. Unsupervised part-of-speech tagging supporting supervised methods. In *Proceedings of RANLP*.

Eric Brill. 1994. Some advances in transformation-based part of speech tagging. In *National Conference on Artificial Intelligence*, pages 722–727.

Jeffrey Dean and Sanjay Ghemawat. 2004. Mapreduce: Simplified data processing on large clusters. In *Sixth Symposium on Operating System Design and Implementation*.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2007. The infinite tree. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 272–279, Prague, Czech Republic, June. Association for Computational Linguistics.

J. Van Gael, Y. Saatci, Y. W. Teh, and Z. Ghahramani. 2008. Beam sampling for the infinite hidden markov model. In *Proceedings of the 25th international conference on Machine learning*, volume 25, Helsinki.

Jianfeng Gao and Mark Johnson. 2008. A comparison of bayesian estimators for unsupervised hidden markov model pos taggers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 344–352.

Sharon Goldwater and Tom Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic, June. Association for Computational Linguistics.

S. Goldwater, T. Griffiths, and M. Johnson. 2006. Interpolating between types and tokens by estimating power-law generators. *Advances in Neural Information Processing Systems*, 18.

Mark Johnson. 2007. Why Doesn't EM Find Good HMM POS-Taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 296–305.

P. Liang, S. Petrov, M. I. Jordan, and D. Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

D. J. C. MacKay. 1997. *Ensemble learning for hidden Markov models*. Technical report, Cavendish Laboratory, University of Cambridge, 1997.

Marina Meilă. 2007. Comparing clusterings—an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 410–420, Prague, Czech Republic, June.

Steven L. Scott. 2002. Bayesian methods for hidden markov models: Recursive computing in the 21st century. *Journal of the American Statistical Association*, 97(457):337–351, March.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Human Language Technology Conference and the 4th Meeting of the North American Association for Computational Linguistics*.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and D. M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In Claire Cardie, Walter Daelemans, Claire Nedellec, and Erik Tjong Kim Sang, editors, *Proceedings of the Fourth Conference on Computational Natural Language Learning*, pages 127–132. Lisbon, Portugal, September.

Andreas Vlachos, Anna Korhonen, and Zoubin Ghahramani. 2009. Unsupervised and Constrained Dirichlet Process Mixture Models for Verb Clustering. In *Proceedings of the EACL workshop on GEometrical Models of Natural Language Semantics*.

A Simple Unsupervised Learner for POS Disambiguation Rules Given Only a Minimal Lexicon

Qiuye Zhao **Mitch Marcus**
Dept. of Computer & Information Science
University of Pennsylvania
qiuye, mitch@cis.upenn.edu

Abstract

We propose a new model for unsupervised POS tagging based on linguistic distinctions between open and closed-class items. Exploiting notions from current linguistic theory, the system uses far less information than previous systems, far simpler computational methods, and far sparser descriptions in learning contexts. By applying simple language acquisition techniques based on counting, the system is given the closed-class lexicon, acquires a large open-class lexicon and then acquires disambiguation rules for both. This system achieves a 20% error reduction for POS tagging over state-of-the-art unsupervised systems tested under the same conditions, and achieves comparable accuracy when trained with much less prior information.

1 Introduction

All recent research on unsupervised tagging, as well as the majority of work on supervised taggers, views POS tagging as a sequential labeling problem and treats all POS tags, both closed- and open-class, as roughly equivalent. In this work we explore a different understanding of the tagging problem, viewing it as a process of first identifying functional syntactic contexts, which are flagged by closed-class items, and then using these functional contexts to determine the POS labels. This disambiguation model differs from most previous work in three ways: 1) it uses different encodings over two distinct domains (roughly open- and closed-class words) with complementary distribution (and so decodes separately); 2) it is deterministic and 3) it is non-lexicalized. By learning disambiguation models for open- and closed- classes separately, we found that the deterministic, rule-based model can be learned from unannotated data by a simple strategy of selecting a rule in each appropriate context with the highest count.

In contrast to this, most previous work on unsupervised tagging (especially for English) concentrates on improving the parameter estimation techniques for training statistical disambiguation models from unannotated data. For example, (Smith&Eisner, 2005) proposes contrastive estimation (CE) for log-linear models (CRF), achieving the current state-of-the-art performance of 90.4%; (Goldwater&Griffiths, 2007) applies a Bayesian approach to improve maximum-likelihood estimation (MLE) for training generative models (HMM). In the main experiments of both of these papers, the disambiguation model is learned, but the algorithms assume a complete knowledge of the lexicon with all possible tags for each word. In this work, we propose making such a large lexicon unnecessary by learning the bulk of the lexicon along with learning a disambiguation model.

Little previous work has been done on this natural and simple idea because the clusters found by previous induction schemes are not in line with the lexical categories that we care about. (Chan, 2008) is perhaps the first with the intention of generating "a discrete set of clusters." By applying similar techniques to (Chan, 2008), which we discuss later, we can generate clusters that closely approximate the central open-class lexical categories, a major advance, but we still require a closed-class lexicon specifying possible tags for these words. This asymmetry in our lexicon acquisition model conforms with our understanding of natural language as structured data over two distinct domains with complementary distribution: open-class (lexical) and closed-class (functional).

Provided with only a closed-class lexicon of 288 words, about 0.6% of the full lexicon, the system acquires a large open-class lexicon and then acquires disambiguation rules for both closed- and

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 688–697,
Singapore, 6-7 August 2009.

open-class words, achieving a tagging accuracy of 90.6% for a 24k dataset, as high as the current state-of-the-art (90.4%) achieved with a complete dictionary. In the test condition where both algorithms are provided with a full lexicon, and are trained and evaluated over the same 96k dataset, we reduce the tagging error by up to 20%.

In Section 2 we explain our understanding of the POS tagging problem in detail and define the notions of functional context and open- and closed-class elements. Then we will introduce our methods for acquiring the lexicon (Section 3) and learning disambiguation models (Section 4, 5 and 6) step by step. Results are reported in Section 7 followed by Section 8 which discusses the linguistic motivation behind this work and the simplicity and efficiency of our model.

2 The Tagging Problem

In most work on both unsupervised and supervised problem, tagging is viewed as a sequential labeling problem. In this work, however, we would like to explore another view on tagging especially considering language as structured data.

The engineering concept of POS tags derives from the linguistic notion of syntactic category which specifies the combinatorial properties of a word in an underlying (syntactic) structure. Given the parse structure for a given word sequence which breaks the input into recursive functional domains such as IP, VP and NP, the POS tag of each word can be directly inferred. Of course, assuming a pre-parsed structure as input to POS tagging is somewhat ridiculous, but it strongly motivates us to highlight the features of structural information for POS tagging. Without resorting to any intermediate representations richer than the input string, we propose for engineering purposes to capture the features of interest for POS tagging by the functional items in language themselves. Then tagging is considered to be a process of identifying the functional contexts (functional items in context) in which the categorical property of the target item can be inferred.

Following ideas in current linguistic theory discussed in Section 8, we observe that the functional categories and some morphological endings serve as markers of the functional domains themselves (discussed above) and sit abstractly at the edge of those domains; the open-class (lexical) items must sit within appropriate functional domains. More specifically, although long distance dependencies are not at all rare, for a token in sequence, we only consider adjacent closed-class words and the verbal categorical feature (but not morphology) as *functional contexts*, the core concept in our disambiguation model.

Our system uses five open-class categories: three basic lexical categories verb, noun and adverb, and two derived Nominal categories (the two kinds of participles in English); and consider all other words not included in those categories to be closed-class items.

Overall, for the task of unsupervised tagging, we use a rule-based disambiguation model containing disambiguation rules conditioned on functional contexts, and the model is learned from unannotated data constrained by much less lexical knowledge than most previous work, namely the closed-class lexicon as introduced below.

2.1 Closed-class Lexicon

A dictionary containing all possible tags for each word is very useful to constrain the unsupervised learning of a POS disambiguation model, and in most previous work, a full lexicon computed from the WSJ corpus (the source of both training and test datasets) is used for both learning and tagging. Since a full lexicon is not a reasonable resource, we aim to limit the required knowledge to functional (closed-class) words only.

It is hard to define functional words in a linguistically strict sense, but this category is close to the notion within the engineering field of NLP of closed-class words, classes of words that are not open for new members. From the engineering point of view, this implies that a closed class has a finite and static number of members, so its members can be listed once and for all.

For English, lists of closed-class categories such as preposition, pronoun or even degree adverb, are obtainable resources, but this is not necessarily the case for other languages. In this paper, we leave the automatic acquisition of a closed-class lexicon for future work. For experiments in this work, we automatically compute a closed-class lexicon from the WSJ treebank 00-24 sections by picking out those words that are labeled predominantly with closed-class tags[1]. For each word selected as a closed-class word, all possible tags encountered

[1]For each word, if the number of instances labeled by closed-class tags is greater than by open-class tags, we select it as a closed-class word.

more than twice in the WSJ corpus are reserved in the closed-class lexicon, so closed-class words may also have open-class tags in our data set, a source of noise in our results. As a core part of language, this closed-class lexicon containing 288 entries, about 0.6% of the full lexicon by types, should be invariant over various genres, which is confirmed in experiments on both WSJ and Brown corpus[2].

2.2 Tagset

The 45 tags in the Penn Tagset (Marcus *et al.*, 2003) contain more information than just basic lexical categories. In recent work on unsupervised learning of POS taggers following (Smith&Eisner, 2005), the Penn tagset is reduced to 17 tags which nicely improves the tagging performance.

Based on our view of POS tags as local markers of underlying syntactic structure, we derive 27 tags from a feature-based analysis of the original Penn tagset. The main principle for reduction is that we collapse any two tags which are not distinguishable by structural features; such features include +/-N, +/-V for predication and +/-wh, +/-en for movement[3]. For example, under our analysis, the tag 'VBG' has the features [+V, +N, -tense, -en], tag 'VBD' [+V, +tense(past), -en], and 'VB' [+V, -tense(finite), -en]. However, since we do not consider the tense feature to be a structural feature, we do not distinguish 'VBD' from 'VB'; since N(ominal) is a structural feature, 'VBG' remains distinct from both 'VBD' and 'VB'. The 27 tags do not cover all cases of ambiguities of closed-class words in the original Penn tagset. Most notably, adjectives are not separated from nouns.

This reduction naturally follows the crucial properties of our disambiguation model. First of all, our model is not lexicalized, so it can only capture basic interactive relations between categories but cannot capture lexical dependencies, which are heavily required to disambiguate 'RP' with 'IN' or 'PDT' with 'DT' (so these two pairs are collapsed). More importantly, the structural information carried by the closed-class items is the key feature of our disambiguation model, but nouns and adjectives are not distinguishable by their structural positions (in NP), so they are not to be distinguished in our tagset[4].

Tagset	#tags	#closed	#open	amb./token
Smith&Eisner	17	7	6	2.07
ThisWork	27	12	6	1.83
Penn	45	15	15	2.74

Table 1: Comparison of tagsets

Category	Open tags	Closed tags
Verbal	VB	...
Nominal	NN, VBN, VBG	DT, CD, PRP($), WDT, WP($)
None	RB	CC, EX, IN, MD, POS, TO

Table 2: N/V categories of 27 POS tags

We use this new reduced tagset with 27 tags in our experiments[5]. For the purposes of comparison, we map the results using our 27 tag tagset to the commonly-used 17 tag tagset[6], and evaluate our algorithms for both tagsets. Table 1 compare the three tagsets, and the ambiguity column shows the average number of ambiguous tags per token in WSJ corpus section 00-24.

2.3 NV category

By using the reduced 27 tags, we found in this work that the heart of the disambiguation task for open-class words is to distinguish them in the Nominal vs. Verbal domains; and for the closed-class words, the Nominal vs. Verbal property of the adjacent context words is also very helpful for

[2]There are two special classes of words worthy of discussion with respect to being closed or open. 1. While the morphological ending '-ly' freely introduces adverbs, this category is otherwise essentially closed class; and 2. There are obviously unboundedly many numbers(CD), but all these match some regular pattern. So we include adverbs without explicit morphological marking in the closed-class lexicon (we frankly doubt adverbs can be acquired by distributional clustering); and as for numbers, we embed exactly such a regular pattern in our model.

[3]Not all features of tags are listed here, and further discussion of the feature-based analysis of the tagset is to be reported in other work. This analysis of tags is motivated by Chomsky.

[4]Due to the indistinguishable roles of adjectives and nouns in Noun Phrase, it is also hard to extract the adjectives from nouns for lexicon acquisition.

[5]For open-class categories, we keep VB (for VB*), NN for (NN*), VBG, VBN and RB (for RB*), and we reduce the JJ* tags to the tag NN and for closed-class tags, we keep almost all the original distinctions, except for two pairs: 'PDT' and 'DT'; 'RP' and 'IN'. Also 'WRB' is reduced to 'RB'.

[6]In our tagset, there are two coarser tags which stand for more than one tag in the 17 tags: 'NN' stands for both 'N' and 'ADJ' and 'IN' for both 'RP' and 'IN'. So to map the output of coarser tags to the finer ones, we need to look up the full-lexicon, since adjectives are not extracted from nouns in the lexicon acquisition process. For a word tagged as 'NN' with a possible tag of 'JJ', if the following word is also tagged as 'NN', then the current 'NN' is mapped to 'JJ'. On the other hand, no action is done for mapping 'IN', so gold 'RP' is always mis-tagged as 'IN' after mapping. If our tagging system outputs a finer tag (e.g. WDT) then it is reduced to the corresponding coarser one (e.g. 'W') in mapping to 17 tags.

disambiguation. The Nominal vs. Verbal property is defined through N/V categories of POS tags, and we list each category containing both closed-class and open-class tags in table 2.

3 Acquiring the open-class lexicon

Not being equipped with a full lexicon, our system takes the closed-class lexicon as given, and automatically computes possible tags, which must be open class, for all other words in the acquisition process as described below. There are five open-class tags in our reduced tagset, as we describe above: 'VBG' and 'VBN' represent two kinds of derived Nominal elements, with corresponding morphological endings attached to the verbal roots; and 'RB' represents the adverbial class into which new words can only be introduced if affixed with the special ending '-ly'. Taking into account this special morphology, we divide our construction of the open-class lexicon into two steps: N/V-Clustering and Morphing. At the N/V-clustering step, we classify the base-forms (roots) of open-class words into two clusters in a sparse feature space. At the Morphing step, we count on the embedded functional elements (i.e. morphology) to derive specific tags for words in each cluster.

3.1 Clustering

Inducing syntactic categories is a language acquisition task on which there has been extensive research, e.g. (Clark, 2003) and (Schütze, 1993), based largely on variants of distributional clustering. In a standard setup of POS clustering, each target word to be clustered, w_i, is represented as a vector, *<count(w_i,C_1), count(w_i,C_2),...,count(w_i,C_m)>*, collecting counts of occurrences of w_i in each context, C_j. Then the chosen algorithm clusters the feature vectors according to similarity.

In previous work, the contextual features are lexical, so the length of a feature vector varies from hundreds to thousands of features. The clustering algorithm then runs over this high-dimensional space, which is computationally quite intensive. Unlike previous work, our system only employs seven features, all functional, to represent target words, and we are paid back by a substantial improvement in efficiency. Each open-class word is represented in the feature space by the following seven component vector: *<left:DT, left:MD, mid:-ϕ, mid:-ed, mid:-ing, right:DT, right:MD>*. The first two values in this vector represent the counts of modal verbs (MD) and determiners (DT) occurring to the left of all forms of a base form; the three values in the middle represent the counts of three possible morphological forms of a word; and the last two values represent the counts of an immediately following MD and DT. This radical reduction of the feature space eliminates any need for sophisticated clustering techniques. For the purpose of convenience, we use a basic k-means clustering algorithm which allows us to specify the number of output clusters (Maffi, 2007).

As is well known, clustering all words in a corpus using distributional clustering results in a high number of clusters. For example, (Schütze, 1993) induces 200 clusters and (Clark, 2003) chooses between 16-128; and most of these induced categories are difficult to associate with a specific POS tag. Chan's recent thesis work (Chan, 2008) provides us with a solution to this problem. In the first pass of Chan's model for unsupervised lexical category induction, verbs are separated from all other categories with a high level of purity; the second pass separates adjectives from nouns by using the categorical results from the first pass as an additional feature[7]. His experiments for a wide range of languages show that the *"restriction to cluster base forms only*[8]*"* is crucial to induce clusters more in line with the definition of the open-class syntactic categories we care about here.

Here, we follow a variant of Chan's approach, grouping words with their base-forms for clustering. For example, we group all occurrences of the transformed (morphological) forms, (*start, starts, starting and started*), in a particular context, C_j, together with the base form *start* to form a single count for $(start, C_j)$, in forming the corresponding feature vectors. Given this, since all inflections of one base form share the same feature vector, all inflections enter into the same class of their base-form. In (Chan, 2008), morphological base forms are the output of a new morphology induction algorithm he develops. Here, we simply extract the base form of a word by stripping three possible forms of endings: *-s*, *-ing* and *-ed*[9].

[7] For simplicity, we don't run a second pass but reduce adjectives to noun.

[8] See p.139 in (Chan, 2008)

[9] This simple strategy, as well as more complex morphological analyzers, cannot deal with irregular verbs, so we list in memory the corresponding 'regular' ending of each irregular verb. For example, we know that the ending of *ran* is '-ed', but we DO NOT know that *ran* is only the past tense

3.2 Morphing

After the clustering step, which we intend to separate the Nominal and Verbal classes, two clusters as desired are induced, but we still need a method to automatically decide which one is which. A trick that works well in practice is simply to pick the smaller class as the Verbal class. These two classes reflect the basic categories of the roots; by a generative mechanism observed in most languages, roots (base-forms) are transformed into derived categories by fusing with functional elements, which surface as the few morphological endings in English.

For all words in the Nominal class, except for those with the ending *-ly*, the only possible tag for each is 'NN', since no finer categories of 'NN' exist in our reduced tagset. On the other hand, for a word with ending *-ly* falling into the N class, we simply assume that its tag must be 'RB', although this assumption may have a few exceptions.

The Verbal class contains all words with verbal roots. There are two specific endings in English serving as morphological markers of derived Nominal categories, -ed and -ing, corresponding to derived categories 'VBN' and 'VBG' respectively. So for each word ending with -ed, we assign two possible tags to it, 'VB'(our reduced form of 'VBD') and 'VBN'; and for each word ending with -ing we assume only one possible tag, 'VBG', although this assumption may systematically introduce tagging error confusing 'VBG' and 'NN'. For example, if the feature vector representing the base-form group *start, starts, started,starting* is classified into the verbal class, then both *starts* and *start* will receive one possible tag 'VB'; *starting* will receive one possible tag 'VBG'; but *started* will receive two possible tags 'VBN' and 'VB'.

As one may notice, *start* and *starts* should have two senses, noun and verb, but the Nominal sense is lost in the Morphing step. For such cases, we introduce a simple supplemental process to compensate for the missing Nominal sense. For a word with the possible tag 'VB' (not 'VBG' or 'VBN') as determined in the Morphing step, if it is ever seen following a determiner in context, another possible tag 'NN' will be assigned to it.

Remember that, as introduced in Sect 2.2, 'VBN', 'VBG' and 'NN' are of category N and 'VB' is of category V. Then for each word in the resulting lexicon, there is maximally one possible tag of it falling in either category N or V, so the category information (N or V) is enough for the disambiguation task, as specified in Section 6.

form of *run*, because the ending '-ed' is ambiguous for both past tense and past participle. The list of irregular verbs is obtained from *http://www.englishpage.com*.

4 Unsupervised Tagging

Taking a dictionary as input, the task of unsupervised tagging is to learn a disambiguation model from unannotated data and apply this model for disambiguating the occurrences of words in context. In this section, we are going to introduce the representation of our disambiguation model first, and then discuss how it affects the system design. In the following two sections, we will describe the algorithms for learning and decoding the language model respectively.

4.1 Disambiguation Model

Again, we view tagging as a process of identifying functional context, from which the proper tagging simply follows. Given this, we represent the language model as a set of disambiguation rules conditioned on functional contexts that predict categorical information, with each rule of the form of $r = (con : cat)$ with con and cat the functional context and categorical information respectively.

In both open- and closed-class domains, given a pair of words (W_l, W_r), the disambiguation rules check the functional property of W_l and predicts the N/V category of W_r. However, in the open-class disambiguation model, con represents closed-class items as well as verbal feature, but in the closed-class disambiguation model, con represents closed-class categories (closed-class POS tags). In disambiguating an open-class word, con is checked against the preceding closed-class word or verbal feature (if any), and cat of the following open-class word is predicted. In disambiguating a closed-class word cw, each possible tag of cw may invoke a rule and each rule will predict a N/V category of the following item; if some rule makes the right prediction, the corresponding tag is assigned to cw. For example, *he:V*, a disambiguation rule for open-class words, says that if an open-class token follows the closed-class item *he*, then the Verbal tag should be assigned to this token. On the other hand, *IN:N*, a disambiguation rule for closed-class words, says that if a closed-class token precedes a Nominal word (open- or

closed- class) in context and has a possible tag of 'IN', then tag it with 'IN'.

This rule-based disambiguation model is deterministic in the sense that for each token in context there is maximally one tag that can be predicted. Not being statistically parameterized, this greedy prediction requires that 1) each rule is deterministic and 2) in each context, only one rule is invoked (which is guaranteed by the selection step introduced in Section 5.2). Moreover, this disambiguation model is non-lexicalized in that it is only conditioned on the functional items in context but not the target word itself.

4.2 System Design

Ideally, we should use closed-class tags in context for disambiguating open-class words because closed-class words are potentially ambiguous; but this would cause a chicken-egg problem. If we did this, then the learning of disambiguation rules for closed-class words requires category information for open-class items and vice versa, but none of the required category information is available from the unannotated data[10]. Thanks to how language works (including principally the low degree of ambiguity of closed-class words), it is good enough practically, as shown by our experiments, to encode the disambiguation model for open-class words using closed-class items without categorical information.

In this way, we can learn the disambiguation model of open-class items from raw data; however, closed-class disambiguation model is better learned after open-class words are disambiguated. Then there are four models in the system for learning and tagging over two distinct domains: Model-LC and Model-LO for learning the disambiguation model of closed- and open-class words respectively; Model-DC and Model-DO for disambiguating closed- and open-class words respectively; and they must be executed in a strict order as follows: Model-LO → Model-DO → Model-LC → Model-DC, as illustrated in Figure 1.

5 Learning Disambiguation Rules

In this section, we describe the learning algorithm used in both Model-LO and Model-LC. Although there is no annotated data available for learning,

[10]Our disambiguation model is not statistically parameterized, so this problem can not be resolved by any kind of parameter estimation technique as in previous work on unsupervised tagging.

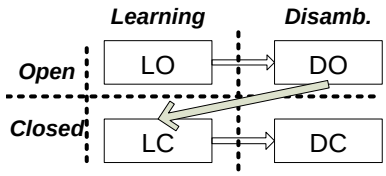


Figure 1: The order of the four models in system.

we can use the unambiguous events in data to establish the disambiguation rules and apply the rules to ambiguous events. The only difference in implementation of the two models lies in the 'rule-extraction', corresponding to different interpretations of unambiguous events for learning open- and closed-class disambiguation models. After being extracted from pairs of adjacent words in the input sequence, the rules are counted and selected using the same algorithm in both models.

5.1 Rule-extraction

For open-class words, disambiguation rules are extracted from raw data. A pair of adjacent words (W_l, W_r) is considered unambiguous if it satisfies the following two conditions: 1. W_l is in the closed class or an unambiguous type with only possible tag of 'VB'; and 2. all possible tags of W_r fall in the same N/V category (Nominal or Verbal but not mixed). If (W_l, W_r) is unambiguous in this sense, then extract rule $r = (con : cat)$, where con is W_l (for closed-class words) or 'V' (for unambiguous verbal words), and cat is the N/V category of W_r. For example, in the sequence *(...he has claimed..)*, the pair *(he, has)* is unambiguous in that *he* is a closed-class item and *has* has only one possible tag, 'VB', so a rule $((he : V)$ is extracted; but *(has, claimed)* is not usable since *claimed* has two possible tags: 'VB' of category V and 'VBN' of category N.

Disambiguation rules for closed-class words are extracted after open-class disambiguation. A pair of adjacent words (W_l, W_r) is considered unambiguous if it satisfies the following two conditions: 1. W_l is in the closed class and has only one possible tag in the closed-class lexicon; 2. W_r is either disambiguated or all possible tags of W_r fall in the same N/V category. If (W_l, W_r) is unambiguous in the above sense, then extract rule $r = (con : cat)$, where con is the single tag of W_l, and cat is the N/V category of W_r. For example, in the sequence *"...for his stepping..."*, the pair *(for his)* is unambiguous in that *for* has only

one possible tag 'IN' and both possible tags of *his*, 'PRP' and 'PRP$', fall into the Nominal category, then a rule $(IN : N)$ is extracted; but *(his about)* is not usable since *his* has more than one possible tag and *about* has two possible tags, 'RB' and 'IN', which are neither both 'N' nor both 'V'.

5.2 Counting and Selecting

In the counting step, a set of rules R is first initialized to be empty, and then, as each disambiguation rule r is generated while passing through the data, if not already in R, it is added with an initial count of one; otherwise, N_r, the count of r, is incremented by one. Note that we know that for a rule, $(con : cat)$, the prediction cat can only be either N or V; then for each context con, there are two forms of rules counted, $(con : N)$ or $(con : V)$. By selecting the rule with a greater count for each context, we guarantee that the resulting disambiguation model is deterministic.

6 Tagging

Given our rule-based, deterministic language model, tagging is a straightforward process of decoding the disambiguation rules. Recall that there are two separate tagging models in the system, Model-DO and Model-DC for disambiguating open- and closed-class respectively.

The inputs to Model-DO are the open-class lexicon, the disambiguation rules learned in Model-LO and raw data in sequence. For each ambiguous open-class word w in sequence if the preceding closed-class word (if any) invokes a disambiguation rule, $r = (con : cat)$, then pick the possible tag of w that falls in the category of cat (N or V), as discussed in Section 3.2. If no rule is triggered our default choice is 'NN'; but if 'NN' is not a possible tag, we assume the default domain is Verbal (so the 'VB' tag is favored).

The application of disambiguation rules in Model-DC is a little more complex. For each ambiguous closed-class word cw in sequence followed by a token of category cat, N or V, pick a possible tag of cw, con, such that $(con : cat)$ is a rule learned in Model-LC. If no tag is picked, a random choice is made. While there are residual cases that no functional context can help with tagging, the disambiguation model proposed here combined with random choice results in a good overall performance, as shown in section 7.3.

	dict. with words of count $> d$				
d (percent lex.)	1 (100%)	2 (55%)	3 (41%)	∞ (0.6%)	#tag -
BHMM2	87.3	79.6	65.0	-	17
CRF/CE	90.4	77.0	71.7	-	17
model-17	**91.8**	...	...	**90.6**	17
model-27	*93.2*	...	...	*92.1*	27
LDA+AC	93.4	91.2	89.7	-	17

Table 3: Tagging accuracy with partial dictionaries over 24k dataset; our closed-class lexicon is the closest approximation to the ∞ column .

7 Results

Our unsupervised tagging system is compared to the following models As reported in (Banko&Moore, 2004), 'the quality of the lexicon made available to unsupervised learner made the greatest difference to tagging accuracy'. So we only compare our experiments to recent work built over the same dataset and a full lexicon automatically extracted from the Penn Treebank. As described in section 2.1, the closed-class lexicon, special in our experiments, is also automatically constructed from the WSJ corpus, and will be used in experiments on both WSJ and Brown corpora below[11]. CRF/CE (Smith&Eisner, 2005) and BHMM2 (Goldwater&Griffiths, 2007) have been discussed briefly in the introduction. LDA+AC (Toutanova&Johnson, 2007) is actually a semi-unsupervised model given the prior on $p(t|w)$; despite this additional information, our model outperforms it in experiments with partial dictionaries. For the purpose of comparison, our experiments use the same dataset as in these previous work, varying in sizes from 12K to 96K. In addition to reporting on our own tagset with 27 tags, we also map the results onto the 17 tags used in other models as explained above.

7.1 Unsupervised Tagging over Partial Dictionaries

As shown in Table 3, reducing the dictionary by filtering rare words (with count$<= d$) has not been a promising track to follow for accomplishing the task with as little information as possible. However, by introducing a lexicon acquisition step, we achieve a tagging accuracy of 90.6% for the 24K test data with no prior open-class lexicon, provided with only a minimal lexicon of closed-class items (about 0.6% of the full lexicon), as high as

[11] If we control the quality of the closed-class lexicon (but still leave the full-lexicon untouched) by filtering out errors in the Treebank, the performance is considerably higher.

size	12K	24k	48k	96K	#tag	lex.
BHMM2	85.8	84.4	85.7	85.8	17	full
CRF/CE	86.2	88.6	88.4	89.4	17	full
Model-17	**91.0**	**91.6**	**91.6**	**91.5**	17	full
Model-27	*93.1*	*93.6*	*93.5*	*93.4*	27	full
model-17	**88.9**	**89.3**	**90.2**	**90.4**	17	closed
model-27	*90.9*	*91.2*	*92.0*	*92.2*	27	closed

Table 4: Tagging Accuracy of models trained over dataset varying in sizes with full/closed-class lexicon

the best previous performance of 90.4 given a full lexicon (CRF/CE with $d = 1$)[12].

One other work that investigates the use of a limited lexicon is (Haghighi&Klein, 2006), which develops a prototype-drive approach to propagate the categorical property using distributional similarity features; using only three exemplars of each tag, they achieve a tagging accuracy of 80.5% using a somewhat larger dataset but also the full Penn tagset, which is much larger.

7.2 Varying in sizes

As shown in Table 4, our new algorithm reduces tagging error by up to 20% over the state-of-the-art given a full lexicon, from 89.4% to 91.5% over the 96k dataset[13].

To better understand the learning property of our system and to get an estimate of the variance of our results above, we repeated the experiments above, starting with either the full lexicon or just the closed-class lexicon, with datasets varying from 0.5K to 96K in size, and repeated each experiment 60 times on different sequences, with four samples randomly selected from the Brown corpus, one from the training data reported above and the others from the WSJ corpus. As shown in Figure 2, for the closed-class lexicon experiments, the standard deviation of tagging accuracy over the dataset of each size sharply decreases as the size of the data increases, as expected. It is also clear that

[12]Since we are facing an unsupervised task, the training set is unannotated, and hence there is no reason not to use it as the test set as well. For the sake of comparison, we use the same split of the dataset for training as previous work. In Table 3 the tagging model is trained over 96k and evaluated on 24k, but in Table 4, the tagging model is trained and evaluated over test and training sets of the same size.

[13]With a full lexicon, we need to disambiguate between open-class tags which fall into the same N/V category, which is beyond the ability of our disambiguation rules which predict N or V only. When more than one possible tag in the same category predicted by the disambiguation rule, we simply make a random choice. Although not as constrained as the acquired lexicon, a full lexicon does improve the tagging performance, since the automatic lexicon acquisition is far from perfect.

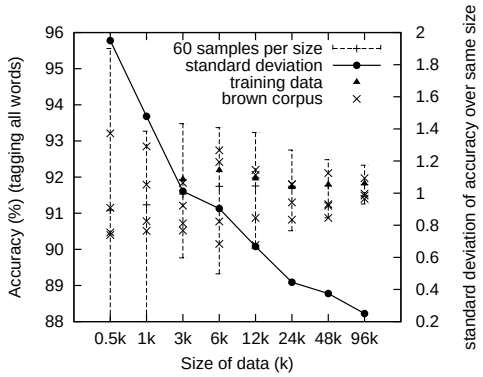


Figure 2: Standard Deviation of Tagging Accuracy with closed-class lexicon; 60 samples for each size, randomly selected from both Brown and WSJ corpus.

	system with closed-class lexicon		system with full lexicon	
sub-model	#errors	accuracy	#errors	accuracy
Model-DO	1089	87.3%	3546	78.9%
Model-DC	1694	89.6%	1709	89.7%
random	1148	44.2%	981	44.9%
recall	3650	-	75	-
total	7581	75.2%	6311	82.1%
#ambiguous	30563		35229	

Table 5: The number of errors and percent *ambiguous* tokens tagged correctly in the 96k dataset with 27 tags. For either system built upon closed-class lexicon or full lexicon, the table shows the disambiguation accuracy and number of errors for each sub-model in the system: Model-DO for disambiguating open-class, Model-DC for disambiguating Closed-class and random choice. The numbers of recall errors (gold tag not in dictionary) and total errors for each system are also shown.

the performance of our algorithm on the Brown corpus is as strong as on the WSJ corpus. Results for the full-lexicon are similar.

7.3 Error Analysis

There are certainly cases that no functional context can help with tagging, since our disambiguation models are encoded by functional context only. Thus it is worth a closer look to how often the system resorts to random choice, as well as to the disambiguation accuracy of either disambiguation model for open- and closed- class learned from unannotated data. We show the disambiguation accuracy of ambiguous words only for each model in Table 5, and also the number of errors due to imperfect lexicons or random choice.

8 Discussion and Future Work

In this work on unsupervised tagging, we combine lexicon acquisition with the learning of a

POS disambiguation model. Moreover, the disambiguation model we used is deterministic, non-lexicalized and defined over two distinct domains with complementary distribution (open- and closed-class).

Building a lexicon based on induced clusters requires our morphological knowledge of three special endings in English: *-ing*, *-ed* and *-s*; on the other hand, to reduce the feature space used for category induction, we utilize vectors of functional features only, exploiting our knowledge of the role of determiners and modal verbs. However, the above information is restricted to the lexicon acquisition model. Taking a lexicon as input, which either consists of a known closed-class lexicon together with an acquired open-class lexicon or is composed by automatic extraction from the Penn Treebank, we need NO language-specific knowledge for learning the disambiguation model.

We would like to point the reader to (Chan, 2008) for more discussion on Category induction[14]; and discussions below will concentrate on the proposed disambiguation model.

Current Chomskian theory, developed in the Minimalist Program (MP) (Chomsky, 2006), argues (very roughly speaking) that the syntactic structure of a sentence is built around a scaffolding provided by a set of functional elements[15]. Each of these provides a large tree fragment (roughly corresponding to what Chomsky calls a *phase*) that provide the piece parts for full utterances. Chomsky observes that when these fragments combine, only the very edge of the fragments can change and that the internal structure of these fragments is rigid (he labels this observation the Phase Impenetrability Condition, PIC). With the belief in PIC, we propose the concept of functional context, in which category property can be determined; also we notice the distinct distribution of the elements (functional) on the edge of *phase* and those (lexical) assembled within the *phase*.

Instead of chasing the highest possible performance by using the strongest method possible, we wanted to explore how well a deterministic, non-lexicalized model, following certain linguistic intuitions, can approach the NLP problem. For the unsupervised tagging task, this simple model, with less than two hundred rules learned, even outperforms non-deterministic generative models with ten of thousands of parameters.

Another motivation for our pursuit of this deterministic, non-lexicalized model is computational efficiency[16]. It takes less than **3 minutes** total for our model to acquire the lexicon, learn the disambiguation model, tag raw data and evaluate the output for a 96k dataset on a small laptop[17]. And a model using only counting and selecting is common in the research field of language acquisition and perhaps more compatible to the way humans process language.

We are certainly aware that our work does not yet address two problems: 1). How the system can be adapted to work for other languages and 2) How to automatically obtain the knowledge of functional elements. We believe that, given the proper understanding of functional elements, our system will be easily adapted to other languages, but we clearly need to test this hypothesis. Also, we are highly interested in completing our system by incorporating the acquisition of functional elements. (Chan, 2008) presents an extensive discussion of his work on morphological induction and (Mintz *et al.*, 2002) presents interesting psychological experiments we can build on to acquire closed-class words.

9 Acknowledgments

We thank the National Science Foundation for its support of this work under grant IIS-0415138. We greatly appreciate the comments of the anonymous reviewers; section 7.3 is newly added and two more paragraphs are added to section 2.2 in response to their comments. Also, we would like to thank an anonymous reviewer of a earlier version of this paper, whose thoughtful suggestion led to a restructuring of the current version. We benefited greatly from our discussions with Dr. Charles Yang. Noah Smith provided the data sets and details of the 17 tag tagset used in previous work. Finally, we thank Constantine Lignos for his careful editing of earlier versions.

[14] In our experiment, using the base-forms and adding a compensation process improves the coverage rate of the acquired lexicon from 79% to 93%.

[15] Such as determiners (for NPs), complementizers like *that* (for clauses), and case assigning elements associated with transitive verbs (for propositions).

[16] In some sense, the Minimalist Program was proposed to explore the idea that the existence of Syntax is especially motivated by efficient language processing.

[17] On a Intel Core 2 Duo P8600 2.40 GHz CPU.

References

Michele Banko and Robert C. Moore. 2004. Part of speech tagging in context. *In COLING, 2004.*

Erwin Chan. 2008. Structures and distributions in morphological learning. *Ph.D. dissertation, Dept. of Computer and Information Science, UPenn.*

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. *In Proceedings of the 10th Meeting of the EACL.*

Chomsky, N. 2006. Approaching UG from below. *MIT.*

Frank, Robert. 2006. Phase theory and Tree Adjoining Grammar. *Lingua.*

Sharon Goldwater and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised Part-of-Speech tagging. *In Proceedings of ACL.*

Haghighi and D. Klein. 2006. Prototype-driven learning for sequence models. *In Proceedings of HLT-NAACL.*

Kroch, A. and Joshi, A. K. 1985. Linguistic Relevance of Tree Adjoining Grammars. *Technical Report MS-CIS-85-18, Department of Computer and Information Science, University of Pennsylvania.*

Charles N. Li, Sandra A. Thompson. Mandarin Chinese: A Functional Reference Grammar *University of California Press, 1989*

Hrafn Loftsson. Tagging Icelandic text: A linguistic rule-based approach *Nordic Journal of Linguistics (2008), 31:47-72 Cambridge University Press*

Leonardo Maffi. Implementation of K-means clustering in Python. *http://www.fantascienza.net/leonardo/so/kmeans/kmeans.html*

Mitchell P. Marcus , Mary Ann Marcinkiewicz , Beatrice Santorini, 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics, v.19 n.2, June 1993.*

T.H. Mintz, E.L. Newport and T.G. Bever. 2002. The distributional structure of grammatical categories in speech to young children. *Cognitive Science 26 (2002), pp. 393C424.*

Hinrich Schütze. 1993. Part-of-speech induction from scratch. *In Proceedings of the 31st Meeting of the ACL.*

Noah A. Smith. Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text. *Ph.D. thesis, Johns Hopkins University Department of Computer Science, Baltimore, MD, October 2006.*

L Shen, G Satta and A Joshi. 2007. Guided Learning for Bidirectional Sequence Classification *In Proceedings of ACL.*

Noah Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. *In Proceedings of the 43rd Meeting of the ACL.*

Kristina Toutanova and Mark Johnson. 2007. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. *In NIPS2007.*

Charles Yang. 2002. Knowledge and learning in natural language. *Oxford University Press. (Chapter 3).*

Tree Kernel-based SVM with Structured Syntactic Knowledge for BTG-based Phrase Reordering

Min Zhang **Haizhou Li**
Institute for Infocomm Research
1 Fusionopolis Way,#21-01 Connexis (South Tower)
Singapore 138632
{mzhang,hli}@i2r.a-star.edu.sg

Abstract

Structured syntactic knowledge is important for phrase reordering. This paper proposes using convolution tree kernel over source parse tree to model structured syntactic knowledge for BTG-based phrase reordering in the context of statistical machine translation. Our study reveals that the structured syntactic features over the source phrases are very effective for BTG constraint-based phrase reordering and those features can be well captured by the tree kernel. We further combine the structured features and other commonly-used linear features into a composite kernel. Experimental results on the NIST MT-2005 Chinese-English translation tasks show that our proposed phrase reordering model statistically significantly outperforms the baseline methods.

1 Introduction

Phrase-based method (Koehn et al., 2003; Och and Ney, 2004; Koehn et al., 2007) and syntax-based method (Wu, 1997; Yamada and Knight, 2001; Eisner, 2003; Chiang, 2005; Cowan et al., 2006; Marcu et al., 2006; Liu et al., 2007; Zhang et al., 2007c, 2008a, 2008b; Shen et al., 2008; Mi and Huang, 2008) represent the state-of-the-art technologies in statistical machine translation (SMT). As the two technologies are complementary in many ways, an interesting research topic is how to combine the strengths of the two methods. Many research efforts have been made to address this issue, which can be summarized into two ideas. One is to add syntax into phrase-based model while another one is to enhance syntax-based model to handle non-syntactic phrases. In this paper, we bring forward the first idea by studying the issue of how to utilize structured syntactic features for phrase reordering in a phrase-based SMT system with BTG (Bracketing Transduction Grammar) constraints (Wu, 1997).

Word and phrase reordering is a crucial component in a SMT system. In syntax-based method, word reordering is implicitly addressed by translation rules, thus the performance is subject to parsing errors to a large extent (zhang et al., 2007a) and the impact of syntax on reordering is difficult to single out (Li et al., 2007). In phrase-based method, local word reordering[1] can be effectively captured by phrase pairs directly while local phrase reordering is explicitly modeled by phrase reordering model and distortion model. Recently, many phrase reordering methods have been proposed, ranging from simple distance-based distortion model (Koehn et al., 2003; Och and Ney, 2004), flat reordering model (Wu, 1997; Zens et al., 2004), lexicalized reordering model (Tillmann, 2004; Kumar and Byrne, 2005), to hierarchical phrase-based model (Chiang, 2005; Setiawan et al., 2007) and classifier-based reordering model with linear features (Zens and Ney, 2006; Xiong et al., 2006; Zhang et al., 2007a; Xiong et al., 2008). However, one of the major limitations of these advances is the structured syntactic knowledge, which is important to global reordering (Li et al., 2007; Elming, 2008), has not been well exploited. This makes the phrase-based method particularly weak in handling global phrase reordering. From machine learning viewpoint (Vapnik, 1995), it is computationally infeasible to explicitly generate features involving structured information in many NLP applica-

[1] This paper follows the term convention of global reordering and local reordering of Li et al. (2007), between which the distinction is solely defined by reordering distance (whether beyond four source words) (Li et al., 2007).

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 698–707,
Singapore, 6-7 August 2009.

tions. For example, one cannot enumerate efficiently all the sub-tree features for a full parse tree. This would be the reason why structured features are not fully utilized in previous statistical feature-based phrase reordering model.

Thanks to the nice property of kernel-based machine learning method that can implicitly explore (structured) features in a high dimensional feature space (Vapnik, 1995), in this paper we propose using convolution tree kernel (Haussler, 1999; Collins and Duffy, 2001) to explore the structured syntactic knowledge for phrase reordering and further combine the tree kernel with other diverse linear features into a composite kernel to strengthen the model's predictive ability. Indeed, using tree kernel methods to mine structured knowledge has shown success in some NLP applications like parsing (Collins and Duffy, 2001), semantic role labeling (Moschitti, 2004; Zhang et al., 2007b), relation extraction (Zhang et al., 2006), pronoun resolution (Yang et al., 2006) and question classification (Zhang and Lee, 2003). However, to our knowledge, such technique still remains unexplored for phrase reordering.

In this paper, we look into the phrase reordering problem in two aspects: 1) how to model and optimize structured features, and 2) how to combine the structured features with other linear features and further integrate them into the log-linear model-based translation framework. Our study shows that: 1) the structured syntactic features are very useful and 2) our kernel-based model can well explore diverse knowledge, including previously-used linear features and the structured syntactic features, for phrase reordering. Our model displays one advantage over the previous work that it is able to utilize the structured syntactic features without the need for extensive feature engineering in decoding a parse tree into a set of linear syntactic features.

To have a more insightful evaluation, we design three experiments with three different evaluation metrics. Experimental results on the NIST MT-2005 Chinese-English translation tasks show that our method statistically significantly outperforms the baseline methods in term of the three different evaluation metrics.

The rest of the paper is organized as follows. Section 2 introduces the baseline method of BTG-based phrase translation method while section 3 discusses the proposed method in detail. The experimental results are reported and discussed in section 4. Finally, we conclude the paper in section 5.

2 Baseline System and Method

We use the MaxEnt-based BTG translation system (Xiong et al., 2006) as our baseline. It is a phrase-based SMT system with BTG reordering constraint. The system uses the BTG lexical translation rules ($A \rightarrow x/y$) to translate the source phrase x into target phrase y, and the BTG merging rules ($A \rightarrow [A, A] | < A, A >$) to combine two neighboring phrases with a straight or inverted order. In the translation model, the BTG lexical rules are weighted with several features, such as phrase translation, word penalty and language models, in a log-linear form. With the BTG constraint, the reordering model Ω is defined on the two neighboring phrases A^1 and A^2 and their order $o \in \{straight, inverted\}$ as follows:

$$\Omega = \mathrm{f}(o, A^1, A^2) \quad (1)$$

In the baseline system, a MaxEnt-based classifier with boundary words of the two neighboring phrases as features is used to model the merging/reordering order. The baseline MaxEnt-based reordering model is formulized as follows:

$$\Omega = p_\theta(o|A^1, A^2) = \frac{exp(\sum_i \theta_i h_i(o, A^1, A^2))}{\sum_o exp(\sum_i \theta_i h_i(o, A^1, A^2))} \quad (2)$$

where the functions $h_i(o, A^1, A^2) \in \{0,1\}$ are model feature functions using the boundary words of the two neighboring phrases as features, and θ_i are feature weights that are trained based on the MaxEnt-based criteria.

3 Tree Kernel-based Phrase Reordering Model

3.1 Kernel-based Classifier Solution to Phrase Reordering

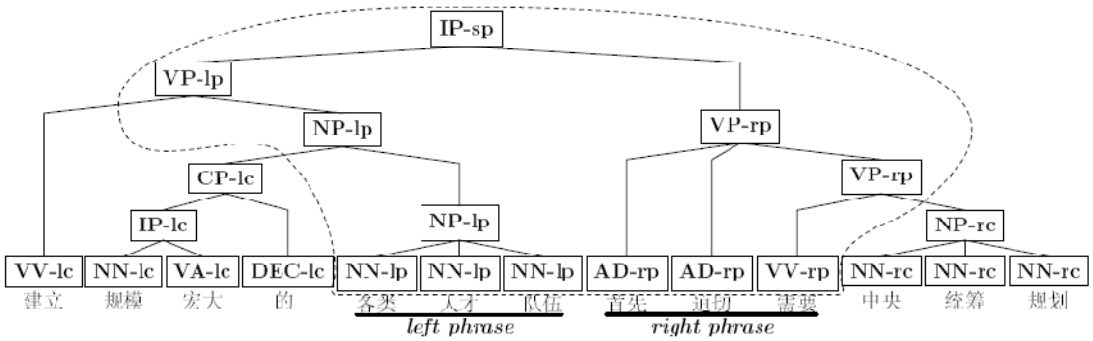
In this paper, phrase reordering is recast as a classification issue as done in previous work (Xiong et al., 2006 & 2008; Zhang et al., 2007a). In training, we use a machine learning algorithm training on the annotated phrase reordering instances that are automatically extracted from word-aligned, source sentence parsed training corpus, to learn a classifier. In testing (decoding), the learned classifier is applied to two adjacent source phrases to decide whether they should be merged (straight) or reordered (inverted) and what their probabilities are, and then these probabilities are used as one feature in the log-linear model in a phrase-based decoder.

In addition to the previously-used linear features, we are more interested in the value of structured syntax in phrase reordering and how to capture it using kernel methods. However, not

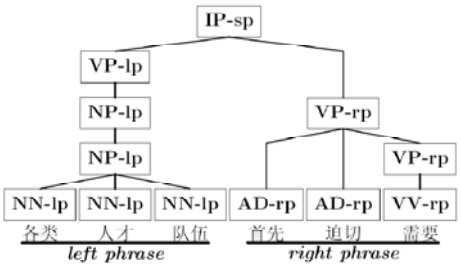
all classifiers are able to work with kernel methods. Only those dot-product-based classifiers can work with kernels by replacing the dot product with a kernel function, where the kernel function is able to directly calculate the similarity between two (structured) objects without enumerating them into linear feature vectors. In this paper, we select SVM as our classifier. In this section, we first define the structured syntactic features and introduce the commonly used linear features, and then discuss how to utilize these features by kernel methods together SVM for phrase reordering

3.2 Structured Syntactic Features

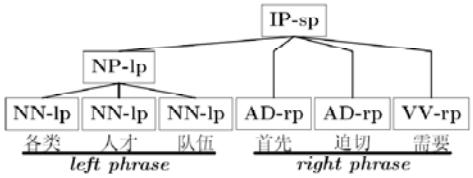
A reordering instance $x = \{A^1, A^2\}$ (see Eq.1) in this paper refers to two adjacent source phrases A^1 and A^2 to be translated. The structured syntactic feature spaces of a reordering instance are defined as the portion of a parse tree of the source sentence that at least covers the span of the reordering instance (i.e. the two neighboring phrases). The syntactic features are defined as all



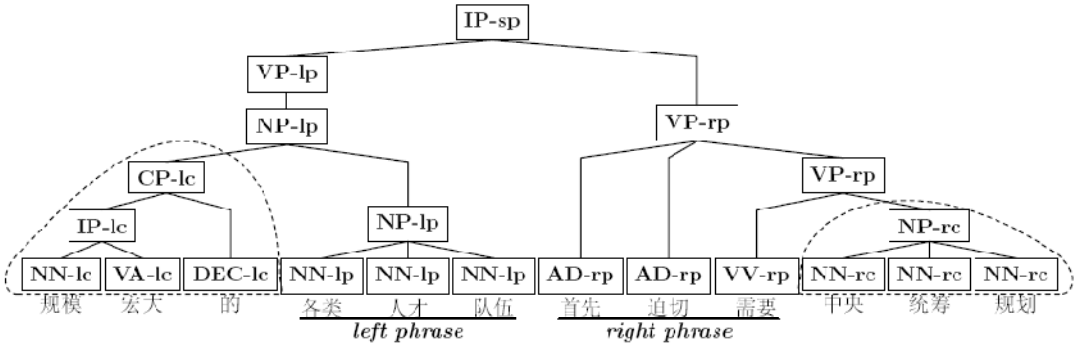
T_1) **Minimum Sub-Tree (MST)**



T_2) **Minimum Sub-Structure (MSS)**



T_4) **Chunking Tree (CT)**



T_3) **Context-sensitive Minimum Sub-Structure (CMSS)**

Figure 1. Different representations of structured syntactic features of a reordering instance in the example sentence excerpted from our training corpus "…建立/build 规模/scale 宏大/mighty 的/of 各类/various types 人才/qualified personnel 队伍/contingent 首先/above all 迫切/urgently 需要/necessary 中央/central authorities 统筹/overall 规划/planning…(To build a mighty contingent of qualified personnel of various types, it is necessary, above all, for the central authorities to make overall planning.) ", where "各类/various types 人才/qualified personnel 队伍/contingent (contingent of qualified personnel of various types)" is the 1st/left phrase and "首先/above all 迫切/urgent 需要/necessary (it is necessary, above all, …)" is the 2nd/right phrase. Note that different function tags are attached to the grammar tag of each internal node.

the possible subtrees in the structured feature spaces. We can see that the structured feature spaces and their features are encapsulated by a full parse tree of source sentences. Thus, it is critical to understand which portion of a parse tree (i.e. structured feature space) is the most effective to represent a reordering instance. Motivated by the work of (Zhang et al., 2006), we here examine four cases that contain different sub-structures as shown in Fig. 1.

(1) Minimum Sub-Tree (MST): the sub-tree rooted by the nearest common ancestor of the two phrases. This feature records the minimum sub-structure covering the two phrases and its left and right contexts as shown in Fig 1.T_1.

(2) Minimum Sub-Structure (MSS): the smallest common sub-structure covering the two phrases. It is enclosed by the shortest path linking the two phrases. Thus, its leaf nodes exactly consist of all the phrasal words.

(3) Context-sensitive Minimum Sub-Structure (CMSS): the **MSS** extending with the 1st left sibling node of the left phrase and the 1st right sibling node of the right phrase and their descendants. If sibling is unavailable, then we move to the parent of current node and repeat the same process until the sibling is available or the root of the **MST** is reached.

(4) Chunking Tree (CT): the base phrase list extracted from the **MSS**. We prune out all the internal structures of the **MSS** and only keep the root node and the base phrase list for generating the chunking tree.

Fig. 1 illustrates the different representations of an example reordering instance. T_1 is the **MST** for the example instance, where the sub-structure circled by a dotted line is the **MSS**, which is also shown in T_2 for clarity. We can see that the **MSS** is a subset of the **MST**. By T_2 we would like to evaluate whether the structured information is effective for phrase reordering while by comparing the system performance when using T_1 and T_2, we would like to evaluate whether the structured context information embedded in the **MST** is useful to phrase reordering. T_3 is the **CMSS**, where the two sub-structures circled by dotted lines are included as the context to T_2 and make T_3 limited context-sensitive. This is to evaluate whether the limited context information in the **CMSS** is helpful. By comparing the performance of T_1 and T_3, we would like to see whether the larger context in T_1 is a noisy feature. T_4 is the **CT**, where only the basic structured information is kept. By comparing the performance of T_2 and T_4, we would like to study whether the high-level structured syntactic features in T_2 are useful to phrase reordering.

After defining the four structured feature spaces, we further partition each feature space into five parts according to their functionalities. Because it only makes sense to evaluate two partitions of the same functionality between two reordering instances, the feature space partition leads to a more precise similarity calculation. As shown in Fig 1, all the internal nodes in each partition are labeled with a unique function tag in the following way:

- **Left Context (-lc):** nodes in this partition do not cover any phrase word and they are all in the left of the left phrase.
- **Right Context (-rc):** nodes in this partition do not cover any phrase word and they are all in the right of the right phrase.
- **Left Phrase (-lp):** nodes in this partition only cover the first phrase and/or its left context.
- **Right Phrase (-rp):** nodes in this partition only cover the second phrase and/or its right context.
- **Shared Part (-sp):** nodes in this partition at least cover both of the two phrases partially.

No lexical word is tagged since it is not a part of the structured features, and therefore not participating in the tree kernel computing.

3.3 Linear Features

In our study, we define the following lexicalized linear features which are easily to be extracted and integrated to our composite kernel:

- Leftmost and rightmost boundary words of the left and right source phrases
- Leftmost and rightmost boundary words of the left and right target phrases
- Internal words of the four phrases (excluding boundary words)
- Target language model (LM) score difference (monotone-inverted)

In total, we arrive at 13 features, including 8 boundary word features, 4 (kinds of) internal word features and 1 LM feature. The first 12 features have been proven useful (Xiong et al., 2006; Zhang et al., 2007a) to phrase reordering. LM score is certainly a strong evidence for modeling word orders and lexical selection. Although it is already used as a standalone feature in the log-linear model, we also would like to explicitly re-optimize it together with other reordering features in our reordering model.

3.4 Tree Kernel, Composite Kernel and Integrating into our Reordering Model

As discussed before, we use convolution tree kernel to capture the structured syntactic feature implicitly by directly computing similarity between the parse-tree representations of two reordering instances with explicitly enumerating all the features one by one. In convolution tree kernel (Collins and Duffy, 2001), a parse tree T is implicitly represented by a vector of integer counts of each sub-tree type (regardless of its ancestors):

$$\phi(T) = (\#\ subtree_1(T), \ldots, \#\ subtree_n(T))$$

where $\#\ subtree_i(T)$ is the occurrence number of the i^{th} sub-tree type ($subtree_i$) in T. Since the number of different sub-trees is exponential with the parse tree size, it is computationally infeasible to directly use the feature vector $\phi(T)$. To solve this computational issue, Collins and Duffy (2001) proposed the following parse tree kernel to calculate the dot product between the above high dimensional vectors implicitly.

$$\begin{aligned} K(T_1,T_2) &= <\phi(T_1),\phi(T_2)> \\ &= \sum_i \left(\left(\sum_{n_1\in N_1} I_{subtree_i}(n_1)\right)\cdot\left(\sum_{n_2\in N_2} I_{subtree_i}(n_2)\right)\right) \\ &= \sum_{n_1\in N_1}\sum_{n_2\in N_2}\Delta(n_1,n_2) \end{aligned}$$

where N_1 and N_2 are the sets of nodes in trees T_1 and T_2, respectively, and $I_{subtree_i}(n)$ is a function that is 1 iff the $subtree_i$ occurs with root at node n and zero otherwise, and $\Delta(n_1,n_2)$ is the number of the common *subtrees* rooted at n_1 and n_2, i.e.,

$$\Delta(n_1,n_2) = \sum_i I_{subtree_i}(n_1)\cdot I_{subtree_i}(n_2)$$

$\Delta(n_1,n_2)$ can be further computed efficiently by the following recursive rules:

Rule 1: if the productions (CFG rules) at n_1 and n_2 are different, $\Delta(n_1,n_2)=0$;

Rule 2: else if both n_1 and n_2 are pre-terminals (POS tags), $\Delta(n_1,n_2)=1\times\lambda$;

Rule 3: else,

$$\Delta(n_1,n_2) = \lambda\prod_{j=1}^{nc(n_1)}(1+\Delta(ch(n_1,j),ch(n_2,j)))$$

where $nc(n_1)$ is the child number of n_1, $ch(n,j)$ is the j^{th} child of node n and λ ($0<\lambda<1$) is the decay factor in order to make the kernel value less variable with respect to the *subtree* sizes. In addition, the recursive **Rule 3** holds because given two nodes with the same children, one can construct common sub-trees using these children and common sub-trees of further offspring. The time complexity for computing this kernel is $O(|N_1|\cdot|N_2|)$ and in practice in near to linear computational time without the need of enumerating all subtree features.

In our study, the linear feature-based similarity is simply calculated using dot-product. We then define the following composite kernel to combine the structured features-based and the linear features-based similarities:

$$K_c(x_1,x_2) = \alpha\cdot K_t(x_1,x_2) + (1-\alpha)\cdot K_l(x_1,x_2) \quad (3)$$

where K_t is the tree kernel over the structured features and K_l is the linear kernel (dot-product) over the linear features. The composite kernel K_c is a linear combination of the two individual kernels, where the coefficient α is set to its default value 0.3 as that in Moschitti (2004)'s implementation. The kernels return the similarities between two reordering instances based on their features used. Our basic assumption is, the more similar the two reordering instances of x_1 and x_2 are, the more chance they share the same order.

Now let us see how to integrate the kernel functions into SVM. The linear classifier learned by SVM is formulized as:

$$f(x) = \mathrm{sgn}(\textstyle\sum_i y_i a_i x\bullet x_i + b) \quad (4)$$

where a_i is the weight of a support vector x_i (i.e., a support reordering instance $x_i = \{A^1, A^2\}$in our study), y_i is its class label (1: $straight$ or -1: $inverted$ in our study) and b is the intercept of the hyperplane. An input reordering instance x is classified as positive (negative) if $f(x)>0$ ($f(x)<0$).

Based on the linear classifier, a kernelized SVM can be easily implemented by simply replacing the dot product $x*x_i$ in Eq (4) with a kernel function $K(x,x_i)$. Thus, the kernelized SVM classifier is formulated as:

$$f(x) = \mathrm{sgn}(\textstyle\sum_i y_i a_i K(x,x_i) + b) \quad (5)$$

where $K(x,x_i)$ is either $K_c(x,x_i)$, $K_t(x,x_i)$ or $K_l(x,x_i)$ in our study. Following Eq (1), our reordering model (implemented by the kernelized SVM) can be formulized as follows:

$$\begin{aligned} \Omega &= \mathrm{f}(o,A^1,A^2) = p_{svm}(o|x=\{A^1,A^2\}) \\ &= sgn(\textstyle\sum_i(y_i a_i K(x,x_i)+b)) \end{aligned} \quad (6)$$

A reordering instance x is classified as *straight* (or *inverted*) if $p_{svm}(o|x)>0$ (or $p_{svm}(o|x)<0$). Eq (6) and Eq (2) show the difference between our kernalized SVM-based reordering

model and the MaxEnt-based reordering model. The main difference between them lies in that our model is able to utilize structured syntactic features by kernalized SVM while the previous work can only use lexicalized word features by MaxEnt-based classifier.

Finally, because the return value of $p_{svm}(o|x)$ is a distance function rather than a probability, we use a sigmoid function to convert $p_{svm}(o|x)$ to a posterior probability as shown using the following to functions and apply it as one feature to the log-linear model in the decoding.

$$P(straight \mid x) = \frac{1}{1+e^{-p_{svm}(o|x)}} \quad \text{and}$$

$$P(inverted \mid x) = \frac{1}{1+e^{p_{svm}(o|x)}}$$

where *straight* represents a positive instance and *inverted* represents a negative instance.

4 Experiments and Discussion

4.1 Experimental Settings

Basic Settings: we evaluate our method on Chinese-English translation task. We use the FBIS corpus as training set, the NIST MT-2002 test set as development (dev) set and the NIST MT-2005 test set as test set. The Stanford parser (Klein and Manning, 2003) is used to parse Chinese sentences on the training, dev and test sets. GIZA++ (Och and Ney, 2004) and the heuristics "grow-diag-final-and" are used to generate *m-to-n* word alignments. The translation model is trained on the FBIS corpus and a 4-gram language model is trained on the Xinhua portion of the English Gigaword corpus using the SRILM Toolkits (Stolcke, 2002) with modified Kneser-Ney smoothing (Kenser and Ney, 1995). For the MER training (Och, 2003), we modify Koehn's MER trainer (Koehn, 2004) to train our system. For significance test, we use Zhang et al's implementation (Zhang et al, 2004).

Baseline Systems: we set three baseline systems: **B1**) Moses (Koehn et al., 2007) that uses lexicalized unigram reordering model to predict three orientations: monotone, swap and discontinuous; **B2**) MaxEnt-based reordering model with lexical boundary word features only (Xiong et al., 2006); **B3**) Linguistically annotated reordering model for BTG-based (LABTG) SMT (Xiong et al., 2008). For Moses, we used the default settings. We build a CKY-style decoder and integrate the corresponding reordering modelling methods into the decoder to implement the 2nd and the 3rd baseline systems and our system. Except reordering models, all the four systems use the same features in translation model, language model and distortion model as Moses in the log-linear framework. We tune the four systems using the strategies as discussed previously in this section.

Reordering Model Training: we extract all reordering instances from the *m-to-n* word-aligned training corpus. The reordering instances include the two source phrases, two target phrases, order label and its corresponding parse tree. We generate the boundary word features from the extracted reordering instances in the same way as discussed in Xiong et al. (2006) and use Zhang's MaxEnt Tools[2] to train a reordering model for the 2nd baseline system. Similarly, we use the algorithm 1 in Xiong et al. (2008) to extract features and use the same MaxEnt Tools to train a reordering model for the 3rd baseline system. Based on the extracted reordering instances, we generate the four structured features and the linear features, and then use the Tree Kernel Tools (Moschitti, 2004) to train our kernel-based reordering model (linear, tree and composite).

Experimental Design and Evaluation Metrics: we design three experiments and evaluate them using three metrics.

Classification-based: in the first experiment, we extract all reordering instances and their features from the dev and test sets, and then use the reordering models trained on the training set to classify (label) those instances extracted from the dev and test sets. In this way, we can isolate the reordering problem from the influence of others, such as translation model, pruning and decoding strategies, to better examine the reordering models' ability and to give analytical insights into the features. Classification Accuracy (CAcc), the percentage of the correctly labeled instances over all trials, is used as the evaluation metric.

Forced decoding[3]-based and normal decoding-based: the two experiments evaluate the reordering models through a real SMT system. The reordering model and the language model are the same in the two experiments. However, in forced decoding, we train two translation models, one using training data only while another using both

[2] http://homepages.inf.ed.ac.uk/s0450736/maxent.html

[3] A normal SMT decoder filters a translation model according to the source sentences, whereas in forced decoding, a translation model is filtered based on both source sentence and target references. In other words, in forced decoding, the decoder is forced to use those phrases whose translations are already in the references.

training, dev and test data. By forced decoding, we aim to isolate the reordering problem from those of OOV and lexical selections resulting from imperfect translation model in the context of a real SMT task. Besides the the case-sensitive BLEU-4 (Papineni et al., 2002) used in the two experiments, we design another evaluation metrics Reordering Accuracy (RAcc) for forced decoding evaluation. RAcc is the percentage of the adjacent word pairs with correct word order[4] over all words in one-best translation results. Similar to BLEU score, we also use the similar Brevity Penalty BP (Papineni et al., 2002) to penalize the short translations in computing RAcc. Finally, please note for the three evaluation metrics, the higher values represent better performance.

Feature Spaces	CAcc (%)	
	Dev	**Test**
Minimum Sub-Tree (**MST**)	89.87	89.92
Minimum Sub-Structure (**MSS**)	87.95	87.88
Context-Sensitive MSS (**CMSS**)	89.11	89.01
Chunking Tree (**CT**)	86.17	86.21
Linear Features (K_l)	90.79	90.46
K_l w/o using LM feature ($K_{l\text{-}LM}$)	84.24	84.06
Composite Kernel (K_c: **MST**+K_l)	92.98	92.67
MST w/o the 5 function tags	86.94	87.03
All are *straight* (*monotonic*)	78.92	78.67

Table 1: Performance of our methods on the dev and test sets with different feature combinations

4.2 Experimental Results

Classification of Instances: Table 1 reports the performance of our defined four structured features, linear feature and the composite kernel. The results are summarized as follows.

The last row reports the performance without using any reordering features. We just suppose that all the translations are monotonic, no reordering happens. The CAccs of 78.92% and 78.67% serve as the bottom line in our study. Compared with the bottom line, the tree kernels over the 4 structured features are very effective for phrase reordering since only structured information is used in the tree kernel[5].

The **CT**s performs the worst among the 4 structured features. This suggests that the middle and high-level structures beyond base phrases are very useful for phrase reordering. The **MSS**s show lower performance than the **CMSS**s and the **MST**s achieve the best performance. This clearly indicates that the structured context information is useful for phrase reordering. For this reason, the subsequent discussions are focused on the **MST**s, unless otherwise specified. The **MSS**s without using the 5 function tags perform much worse than the original ones. This suggests that the partitions of the structured feature spaces are very helpful, which can effectively avoid the undesired matching between partitions of different functionalities. Comparison of K_l and $K_{l\text{-}LM}$ shows the LM plays an important role in phrase reordering. The composite kernel (K_c) performs much better than the two individual kernels. This suggests that the structured and linear features are complementary and the composite kernel can well integrate them for phrase reordering.

Methods	CAcc (%)	
	Dev	**Test**
Minimum Sub-Tree (**MST**)	89.87	89.92
Linear Features (K_l)	90.79	90.46
Composite Kernel (K_c: **MST**+K_l)	92.98	92.67
MaxEnt+boundary word (**B2**)	88.33	86.97
MaxEnt+linguistic features (**B3_1**)	84.83	83.92
MaxEnt+LABTG (**B3: B2+ B3_1**)	88.82	88.18

Table 2: Performance comparison of different methods

Table 2 compares the performance of the baseline methods with ours. Comparison between **B3_1** and **MST** clearly demonstrates that the structured syntactic features are much more effective than the linear syntactic features that are manually extracted via heuristics. It also suggests that the tree kernel can well capture the structured features implicitly. K_l outperforms **B2**. This is mainly due to the contribution of LM features. **B2** (MaxEnt-based) significantly outperforms $K_{l\text{-}LM}$ in Table 1 (SVM-based). This suggests that phrase reordering may not be a good linearly binary-separable task if only boundary word features are used. Our composite kernel (K_c) significantly outperforms LABTG (**B3**). This mainly

[4] An adjacent word pair $w_i w_{i+1}$ in a translation have correct word order if and only if w_i appears before w_{i+1} in translation references. Note than the two words may not be adjacent in the references even if they have correct word order.

[5] The tree kernel algorithm only compares internal structures. It does not concern any lexical leaf nodes.

attributes to the contributions of structured syntactic features, LM and the tree kernel.

Forced Decoding: Table 3 compares the performance of our composite kernel with that of the LABTG (Baseline 3) in forced decoding. As discussed before, here we try two translation models.

The composite kernel outperforms the LABTG in all test cases. This further validates the effectiveness of the kernel methods in phrase reordering. There are still around 30% words reordered incorrectly even if we use the translation model trained on both training, dev and test sets. This reveals the limitations of current SMT modeling methods and suggests interesting future work in this area. The source language OOV[6] rate in forced decoding (13.6%) is much higher that in normal decoding (6.22%, see table 4). This is mainly due to the fact that the phrase table in forced decoding is filtered out based on both source and target languages while in normal decoding it is based on source language only. As a result, more phrases are filtered out in the forced decoding. There is 1.4% OOV even if the translation model is trained on the test set. This is due to the incorrect word alignment, large-span word alignment and different English tokenization strategies used in BLEU-scoring tool and ours.

Methods	Test Set (%)		
	RAcc	OOV	BLEU
Composite Kernel (K_c)	51.03	13.6	38.56
+translation model on Training, dev and test	72.67	1.41	62.87
MaxEnt+LABTG (**B3**)	48.96	13.6	37.32
+translation model on training, dev and test	71.45	1.41	62.14

Table 3: Performance comparison of forced decoding

Methods	Test Set	
	BLEU(%)	OOV(%)
Composite Kernel (K_c)	27.65	6.26
Moses (**B1**)	25.71	6.17
MaxEnt+boundary word(**B2**)	25.99	6.22
MaxEnt+LABTG (**B3**)	26.63	6.22

Table 4: Performance comparison

[6] OOV means a source words has no any English translation according to the translation model. OOV rate is the percentage of the number of OOV words over all the source words.

Normal Decoding/Translation: Table 4 reports the translation performance of our system and the three baseline systems.

Moses (**B1**) and the MaxEnt-based boundary word model (**B2**) achieve comparable performance. This means the lexicalized orientation-based reordering model in Moses performs similarly to the boundary word-based reordering model since the two models are both lexical word-based. However, theoretically, the MaxEnt-based model may suffer less from data sparseness issue since it does not depends on internal phrasal words and uses MaxEnt to optimize feature weights while the orientation-based model uses relative frequency of the entire phrases to compute the posterior probabilities. s. The MaxEnt-based LABTG model significantly outperforms (p<0.05) the MaxEnt-based boundary word model and the lexicalized orientation-based reordering model. This indicates that the linearly linguistically syntactic information is a useful feature to phrase reordering.

Our composite kernel-based model significantly outperforms (p<0.01) the three baseline methods. This again proves that the structured syntactic features are much more effective than the linear syntactic features for phrase reordering and the tree kernel method can well capture the informative structured features. The four methods show very slight difference in OOV rates. This is mainly due to the difference in implementation detail, such as different OOV penalties and other pruning thresholds.

5 Conclusion and Future Work

Structured syntactic knowledge is very useful to phrase reordering. This paper provides insights into how the structured feature can be used for phrase reordering. In previous work, the structured features are selected manually by heuristics and represented by a linear feature vector. This may largely compromise the contribution of the structured features to phrase reordering. Thanks to the nice properties of kernel-based learning method and SVM classifier, we propose leveraging on the kernelized SVM learning algorithm to address the problem. Specifically, we propose using convolution tree kernel to capture the structured features and design a composite kernel to combine the structured features and other linear features for phrase reordering. The tree kernel is able to directly take the structured reordering instances as inputs and compute their similarities without enumerating them into a set of liner

features. In addition, we also study how to find the optimal structured feature space and how to partition the structured feature spaces according to their functionalities. Finally, we evaluate our method on the NIST MT-2005 Chinese-English translation tasks. To provide insights into the model, we design three kinds of experiments together with three different evaluation metrics. Experimental results show that the structured features are very effective and our composite kernel can well capture both the structured and the linear features without the need for extensive feature engineering. It also shows that our method significantly outperforms the baseline methods.

The tree kernel-based phrase reordering method is not only applicable to adjacent phrases. It is able to work with any long phrase pairs with gap of any length in-between. We will study this case in the near future. We would also like to use one individual tree kernel for one partition in a structured feature space. In doing so, we are able to give different weights to different partitions according to their functionalities and contributions. Note that these weights can be automatically tuned and optimized easily against a dev set.

References

David Chiang. 2005. *A hierarchical phrase-based model for SMT*. ACL-05. 263-270.

Michael Collins and N. Duffy. 2001. *Convolution Kernels for Natural Language*. NIPS-2001.

M. R. Costa-jussà and J.A.R. Fonollosa. 2006. *Statistical Machine Reordering*. EMNLP-06. 70-76.

Brooke Cowan, Ivona Kucerova and Michael Collins. 2006. *A discriminative model for tree-to-tree translation*. EMNLP-06. 232-241.

Jason Eisner. 2003. *Learning non-isomorphic tree mappings for MT*. ACL-03 (companion volume).

Jakob Elming. 2008. *Syntactic Reordering Integrated with Phrase-Based SMT*. COLING-08. 209-216.

Michel Galley and Christopher D. Manning. 2008. *A Simple and Effective Hierarchical Phrase Reordering Model*. EMNLP-08. 848-856.

David Haussler. 1999. *Convolution Kernels on Discrete Structures*. TR UCS-CRL-99-10.

T. Joachims. 1998. *Text Categorization with SVM: learning with many relevant features*. ECML-98.

Dan Klein and Christopher D. Manning. 2003. *Accurate Unlexicalized Parsing*. ACL-03. 423-430.

Reinhard Kenser and Hermann Ney. 1995. *Improved backing-off for M-gram language modeling*. ICASSP-95, 181-184

Philipp Koehn, F. Och and D. Marcu. 2003. *Statistical phrase-based translation*. HLT-NAACL-03.

Philipp Koehn, H. Hoang, A. Birch, C. C.-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin and E. Herbst. 2007. *Moses: Open Source Toolkit for SMT*. ACL-07 (poster). 77-180.

Shankar Kumar and William Byrne. 2005. *Local Phrase Reordering Models for Statistical Machine Translation*. HLT-EMNLP-2005. 161-168.

Chi-Ho Li, Dongdong Zhang, Mu Li, Ming Zhou, Minghui Li and Yi Guan. 2007. *A Probabilistic Approach to Syntax-based Reordering for Statistical Machine Translation*. ACL-07. 720-727.

Yang Liu, Yun Huang, Qun Liu and Shouxun Lin. 2007. *Forest-to-String Statistical Translation Rules*. ACL-07. 704-711.

Daniel Marcu, W. Wang, A. Echihabi and K. Knight. 2006. *SPMT: SMT with Syntactified Target Language Phrases*. EMNLP-06. 44-52.

Haitao Mi and Liang Huang. 2008. *Forest-based Translation Rule Extraction*. EMNLP-08. 206-214.

Alessandro Moschitti. 2004. *A Study on Convolution Kernels for Shallow Semantic Parsing*. ACL-04.

Masaaki Nagata, Kuniko Saito, Kazuhide Yamamoto and Kazuteru Ohashi. 2006. *A Clustered Global Phrase Reordering Model for Statistical Machine Translation*. COLING-ACL-06. 713-720.

Franz J. Och and Hermann Ney. 2002. *Discriminative training and maximum entropy models for statistical machine translation*. ACL-02. 295-302.

Franz J. Och. 2003. *Minimum error rate training in statistical machine translation*. ACL-03. 160-167.

Franz J. Och and H. Ney. 2003. *A Systematic Comparison of Various Statistical Alignment Methods*. Computational Linguistics, 29(1):20-51.

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

Kishore Papineni, S. Roukos, T. and W. Zhu. 2002. *BLEU: a method for automatic evaluation of machine translation*. ACL-02. 311-318.

Hendra Setiawan, Min-Yen Kan and Haizhou Li. 2007. *Ordering Phrases with Function Words*. ACL-07. 712-719.

Libin Shen, Jinxi Xu and Ralph Weischedel. 2008. *A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model*. ACL-HLT-08. 577-585.

Andreas Stolcke. 2002. *SRILM - an extensible language modeling toolkit*. ICSLP-02. 901-904.

Christoph Tillmann. 2004. *A Unigram Orientation Model for Statistical Machine Translation*. HLT-NAACL-04 (short paper).

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

Chao Wang, M. Collins and P. Koehn. 2007. *Chinese Syntactic Reordering for Statistical Machine Translation*. EMNLP-CONLL-07. 734-745.

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

Fei Xia and Michael McCord. 2004. *Improving a Statistical MT System with Automatically Learned Rewrite Patterns*. COLING-04.

Deyi Xiong, Qun Liu and Shouxun Lin. 2006. *Maximum Entropy Based Phrase Reordering Model for SMT*. COLING-ACL-06. 521−528.

Deyi Xiong, Min Zhang, Aiti Aw and Haizhou Li. 2008. *A Linguistically Annotated Reordering Model for BTG-based Statistical Machine Translation*. ACL-HLT-08 (short paper). 149-152.

Kenji Yamada and K. Knight. 2001. *A syntax-based statistical translation model*. ACL-01. 523-530.

Xiaofeng Yang, Jian Su and Chew Lim Tan. 2006. *Kernel-Based Pronoun Resolution with Structured Syntactic Knowledge*. COLING-ACL-06. 41-48.

Richard Zens, H. Ney, T. Watanabe and E. Sumita. 2004. *Reordering Constraints for Phrase-Based Statistical Machine Translation*. COLING-04.

Richard Zens and Hermann Ney. 2006. Discriminative Reordering Models for Statistical Machine Translation. WSMT-2006.

Dell Zhang and W. Lee. 2003. *Question classification using support vector machines*. SIGIR-03.

Min Zhang, Jie Zhang, Jian Su and GuoDong Zhou. 2006. *A Composite Kernel to Extract Relations between Entities with Both Flat and Structured Features*. COLING-ACL-06. 825-832.

Dongdong Zhang, M. Li, C.H. Li and M. Zhou. 2007a. *Phrase Reordering Model Integrating Syntactic Knowledge for SMT*. EMNLP-CONLL-07. 533-540.

Min Zhang, W. Che, A. Aw, C. Tan, G. Zhou, T. Liu and S. Li. 2007b. *A Grammar-driven Convolution Tree Kernel for Semantic Role Classification*. ACL-07. 200-207.

Min Zhang, Hongfei Jiang, Ai Ti Aw, Jun Sun, Sheng Li and Chew Lim Tan. 2007c. *A Tree-to-Tree Alignment-based Model for Statistical Machine Translation*.MT-Summit-07. 535-542

Min Zhang, Hongfei Jiang, Ai Ti Aw, Haizhou Li, Chew Lim Tan and Chew Lim Tan and Sheng Li. 2008a. *A Tree Sequence Alignment-based Tree-to-Tree Translation Model*. ACL-HLT-08. 559-567.

Min Zhang, Hongfei Jiang, Haizhou Li, Aiti Aw, Sheng Li. 2008b. *Grammar Comparison Study for Translational Equivalence Modeling and Statistical Machine Translation*. COLING-08. 1097-1104

Ying Zhang, Stephan Vogel and Alex Waibel. 2004. *Interpreting BLEU/NIST scores: How much improvement do we need to have a better system?* LREC-04. 2051-2054.

Discriminative Corpus Weight Estimation for Machine Translation

Spyros Matsoukas and **Antti-Veikko I. Rosti** and **Bing Zhang**
BBN Technologies, 10 Moulton Street, Cambridge, MA 02138
{smatsouk,arosti,bzhang}@bbn.com

Abstract

Current statistical machine translation (SMT) systems are trained on sentence-aligned and word-aligned parallel text collected from various sources. Translation model parameters are estimated from the word alignments, and the quality of the translations on a given test set depends on the parameter estimates. There are at least two factors affecting the parameter estimation: domain match and training data quality. This paper describes a novel approach for automatically detecting and down-weighing certain parts of the training corpus by assigning a weight to each sentence in the training bitext so as to optimize a discriminative objective function on a designated tuning set. This way, the proposed method can limit the negative effects of low quality training data, and can adapt the translation model to the domain of interest. It is shown that such discriminative corpus weights can provide significant improvements in Arabic-English translation on various conditions, using a state-of-the-art SMT system.

1 Introduction

Statistical machine translation (SMT) systems rely on a training corpus consisting of sentences in the source language and their respective reference translations to the target language. These parallel sentences are used to perform automatic word alignment, and extract translation rules with associated probabilities. Typically, a parallel training corpus is comprised of collections of varying quality and relevance to the translation problem of interest. For example, an SMT system applied to broadcast conversational data may be trained on a corpus consisting mostly of United Nations and newswire data, with only a very small amount of in-domain broadcast news/conversational data. In this case, it would be desirable to down-weigh the out-of-domain data relative to the in-domain data during the rule extraction and probability estimation. Similarly, it would be good to assign a lower weight to data of low quality (e.g., poorly aligned or incorrectly translated sentences) relative to data of high quality.

In this paper, we describe a novel discriminative training method that can be used to estimate a weight for each sentence in the training bitext so as to optimize an objective function – expected translation edit rate (TER) (Snover et al., 2006) – on a held-out development set. The training bitext typically consists of millions of (parallel) sentences, so in order to ensure robust estimation we express each sentence weight as a function of sentence-level features, and estimate the parameters of this mapping function instead. Sentence-level features may include the identifier of the collection or genre that the sentence belongs to, the number of tokens in the source or target side, alignment information, etc. The mapping from features to weights can be implemented via any differentiable function, but in our experiments we used a simple perceptron. Sentence weights estimated in this fashion are applied directly to the phrase and lexical counts unlike any previously published method to the author's knowledge. The tuning framework is developed for phrase-based SMT models, but the tuned weights are also applicable to the training of a hierarchical model. In cases where the tuning set used for corpus weight estimation is a close match to the test set, this method yields significant gains in TER, BLEU (Papineni et al., 2002), and METEOR (Lavie and Agarwal, 2007) scores over a state-of-the-art hierarchical baseline.

The paper is organized as follows. Related work on data selection, data weighting, and model adaptation is presented in Section 2. The corpus weight

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 708–717,
Singapore, 6-7 August 2009.

approach and estimation algorithm are described in Section 3. Experimental evaluation of the approach is presented in Sections 4 and 5. Section 6 concludes the paper with a few directions for future work.

2 Related Work

Previous work related to corpus weighting may be split into three categories: data selection, data weighting, and translation model adaptation. The first two approaches may improve the quality of the word alignment and prevent phrase-pairs which are less useful for the domain to be learned. The model adaptation, on the other hand, may boost the weight of the more relevant phrase-pairs or introduce translations for unseen source phrases.

Resnik and Smith (2003) mined parallel text from the web using various filters to identify likely translations. The filtering may be viewed as a data selection where poor quality translation are discarded before word alignment. Yasuda et al. (2008) selected subsets of an existing parallel corpus to match the domain of the test set. The discarded sentence pairs may be valid translations but they do not necessarily improve the translation quality on the test domain. Mandal et al. (2008) used active learning to select suitable training data for human translation. Hildebrand et al. (2005) selected comparable sentences from parallel corpora using information retrieval techniques.

Lu et al. (2007) proposed weighting comparable portions of the parallel text before word alignment based on information retrieval. The relevant portions of the parallel text were given a higher integer weight in GIZA++ word alignment. Similar effect may be achieved by replicating the relevant subset in the training data.

Lu et al. (2007) also proposed training adapted translation models which were interpolated with a model trained on the entire parallel text. Snover et al. (2008) used cross-lingual information retrieval to identify possible bias-rules to improve the coverage on the source side. These rules may cover source phrases for which no translations were learned from the available parallel text.

Koehn and Schroeder (2007) described a procedure for domain adaptation that was using two translation models in decoding, one trained on in-domain data and the other on out-of-domain data. Phrase translation scores from the two models where combined in a log-linear fashion, with weights estimated based on minimum error rate training (Och, 2003) on a designated tuning set.

The method described in this paper can also be viewed as data filtering or (static) translation adaptation, but it has the following advantages over previously published techniques:

1. The estimated corpus weights are *discriminative* and are computed so as to directly optimize an MT performance metric on a pre-defined development set. Unlike the domain adaptation technique in (Koehn and Schroeder, 2007), which also estimates the adaptation parameters discriminatively, our proposed method does not require a manual specification of the in-domain and out-of-domain training data collections. Instead, it automatically determines which collections are most relevant to the domain of interest, and increases their weight while decreasing the weight assigned to less relevant collections.

2. All sentences in the parallel corpus can influence the translation model, as opposed to filtering/discarding data. However, the proposed method can still assign very low weights to parts of the corpus, if it determines that it helps improve MT performance.

3. The framework used for estimating the corpus weights can be easily extended to support discriminative alignment link-level weights, thus allowing the system to automatically identify which portions of the training sentences are most useful.

Naturally, as with any method, the proposed technique has certain limitations. Specifically, it is only concerned with influencing the translation rule probabilities via the corpus weights; it does not change the set of rules extracted. Thus, it is unable to add new translation rules as in Snover et al. (2008). Also, it can potentially lead to parameter over-fitting, especially if the function that maps sentence features to weights is complex and based on a large number of parameters, or if the development set used for estimating the mapping function does not match the characteristics of the test set.

3 Corpus Weights Estimation

3.1 Feature Extraction

The purpose of feature extraction is to identify, for each sentence in the parallel training data, a set of features that can be useful in estimating a weight that is correlated with quality or relevance to the MT task at hand. Starting from sentence-aligned, word-aligned parallel training data, one could extract various types of sentence-level features. For example, we could specify features that describe the two sides of the parallel data or the alignment between them, such as collection id, genre id, number of source tokens, number of target tokens, ratio of number of source and target tokens, number of word alignment links, fraction of source tokens that are unaligned, and fraction of target tokens that are unaligned. Additionally, we could include information retrieval (IR) related features that reflect the relevance of a training sentence to the domain of interest, e.g., by measuring vector space model (VSM) distance of the sentence to the current tuning set, or its log likelihhod with respect to an in-domain language model.

Note that the collection and genre identifiers (ids) mentioned above are bit vectors. Each collection in the training set is mapped to a number. A collection may consist of sentences from multiple genres (e.g., newswire, web, broadcast news, broadcast conversations). Genres are also mapped to a unique number across the whole training set. Then, given a sentence in the training bitext, we can extract a binary vector that contains two non-zero bits, one indicating the collection id, and another denoting the genre id.

It is worth mentioning that in the experiments reported later in this paper we made use of only the collection and genre ids as features, although the framework supports general sentence-level features.

3.2 Mapping Features to Weights

As mentioned previously, one way to map a feature vector to a weight is to use a perceptron. A multi-layer neural network may also be used, but at the expense of slower training. In this work, all of the experiments carried out made use of a perceptron mapping function. However, it is also possible to cluster the training sentences into classes by training a Gaussian mixture model (GMM) on their respective feature vectors[1]. Then, given a feature vector we can compute the (posterior) probability that it was generated by one of the *N* Gaussians in the GMM, and use this N-dimensional vector of posteriors as input to the perceptron. This is similar to having a neural network with a static hidden layer and Gaussian activation functions.

Given the many choices available in mapping features to weights, we will describe the mapping function in general terms. Let $\mathbf{f}_i$ be the $n \times 1$ feature vector corresponding to sentence i. Let $\phi(\mathbf{x}; \boldsymbol{\lambda})$ denote a function $\mathbb{R}^n \rightarrow (0, 1)$ that is parameterized in terms of the parameter vector $\boldsymbol{\lambda}$ and maps a feature vector $\mathbf{x}$ to a scalar weight in $(0, 1)$. The goal of the automatic corpus weight estimation procedure is to estimate the parameter vector $\boldsymbol{\lambda}$ so as to optimize an objective function on a development set.

3.3 Training with Weighted Corpora

Once the sentence features have been mapped to weights, the translation rule extraction and probability estimation can proceed as usual, but with weighted counts. For example, let $w_i = \phi(\mathbf{f}_i; \boldsymbol{\lambda})$ be the weight assigned to sentence i. Let (s, t) be a source-target phrase pair that can be extracted from the corpus, and $\mathcal{A}(s)$ and $\mathcal{B}(t)$ indicating the sets of sentences that s and t occur in. Then,

$$P(s|t) = \frac{\sum_{j \in \mathcal{A}(s) \cap \mathcal{B}(t)} w_j c_j(s, t)}{\sum_{j \in \mathcal{B}(t)} w_j c_j(t)} \qquad (1)$$

where $c_j(\cdot)$ denotes the number of occurrences of the phrase (or phrase-pair) in sentence j.

3.4 Optimizing the Mapping Function

Estimation of the parameters $\boldsymbol{\lambda}$ of the mapping function ϕ can be performed by directly optimizing a suitable objective function on a development set. Ideally, we would like to estimate the parameters of the mapping function so as to directly optimize an automatic MT performance evaluation metric, such as TER or BLEU on the full translation search space. However, this is extremely computationally intensive for two reasons: (a) optimizing in the full translation search space requires a new decoding pass for each iteration of optimization; and (b) a direct optimization of TER or

[1] Note that in order to train such a GMM it may be necessary to first apply a decorrelating, dimensionality reducing, transform (e.g., principal component analysis) to the features.

BLEU requires the use of a derivative free, slowly converging optimization method such as MERT (Och, 2003), because these objective functions are not differentiable.

In our case, for every parameter vector update we need to essentially retrain the translation model (reestimate the phrase and lexical translation probabilities based on the updated corpus weights), so the cost of each iteration is significantly higher than in a typical MERT application. For these reasons, in this work we chose to minimize the expected TER over a translation N-best on a designated tuning set, which is a continuous and differentiable function and can be optimized with standard gradient descent methods in a small number of iterations. Note, that using expected TER is not the only option here; any criterion that can be expressed as a continuous function of the phrase or lexical translation probabilities can be used to optimize $\boldsymbol{\lambda}$.

Given an N-best of translation hypotheses over a development set of S sentences, we can define the expected TER as follows

$$\mathcal{T} = \frac{\sum_{s=1}^{S} \sum_{j=1}^{N_s} p_{sj} \epsilon_{sj}}{\sum_{s=1}^{S} r_s} \tag{2}$$

where N_s is the number of translation hypotheses available for segment s; ϵ_{sj} is the minimum raw edit distance between hypothesis j of segment s (or h_{sj}, for short) and the reference translation(s) corresponding to segment s; r_s is the average number of reference translation tokens in segment s, and p_{sj} is the posterior probability of hypothesis h_{sj} in the N-best. The latter is computed as follows

$$p_{sj} = \frac{e^{\gamma L_{sj}}}{\sum_{k=1}^{N_s} e^{\gamma L_{sk}}} \tag{3}$$

where L_{sj} is the total log likelihood of hypothesis h_{sj}, and γ is a tunable scaling factor that can be used to change the dynamic range of the likelihood scores and hence the distribution of posteriors over the N-best. The hypothesis likelihood L_{sj} is typically computed as a dot product of a decoding weight vector and a vector of various "feature" scores, such as log phrase translation probability, log lexical translation probability, log n-gram language model probability, and number of tokens in the hypothesis. However, in order to simplify this presentation we will assume that it contains a single translation model score, the log phrase translation probability of source given target. This score is a sum of log conditional probabilities, similar to the one defined in Equation 1. Therefore, L_{sj} is indirectly a function of the training sentence weights.

In order to minimize the expected TER $\mathcal{T}$, we need to compute the derivative of $\mathcal{T}$ with respect to the mapping function parameters $\boldsymbol{\lambda}$. Using the chain rule, we get equations (4)-(8), where the summation in Equation 6 is over all source-target phrase pairs in the derivation of hypothesis h_{sm}, ξ is the decoding weight assigned to the log phrase translation score, and the summation in Equation 7 is over all training sentences[2].

Thus, in order to compute the derivative of the objective function we first need to calculate $\frac{\partial \ln P(s_k|t_k)}{\partial \boldsymbol{\lambda}}$ for every phrase pair (s_k, t_k) in the translation N-best based on Equations 7 and 8, which requires time proportional to the number of occurrences of these phrases in the parallel training data. After that, we can compute $\frac{\partial L_{sm}}{\partial \boldsymbol{\lambda}}$ for each hypothesis h_{sm}, based on Equation 6. Finally, we calculate $\frac{\partial \ln p_{sj}}{\partial \boldsymbol{\lambda}}$ and $\frac{\partial \mathcal{T}}{\partial \boldsymbol{\lambda}}$ based on Equations 5 and 4, respectively.

3.5 Implementation Issues

In our system, the corpus weights were trained based on N-best translation hypotheses generated by a phrase-based MT system on a designated tuning set. Each translation hypothesis in the N-best has a score that is a (linear) function of the following log translation probabilities: target phrase given source phrase, source phrase given target phrase, and lexical smoothing term. Additionally, each hypothesis specifies information about its derivation, i.e., which source-target phrase pairs it consists of. Therefore, given an N-best, we can identify the set of unique phrase pairs and use this information in order to perform a filtered accumulation of the statistics needed for calculating the derivative in Equation 8. This reduces the storage needed for the sufficient statistics significantly.

Minimization of the expected TER of the N-best hypotheses was performed using the limited-memory BFGS algorithm (Liu and Nocedal, 1989). Typically, the parameter vector $\boldsymbol{\lambda}$ required about 30 iterations of LBFGS to converge.

Since the N-best provides only a limited representation of the MT hypothesis search space, we regenerated the N-best after every 30 iterations

[2] In the general case where L_{sj} includes other translation scores, e.g., lexical translation probabilities, the derivative $\frac{\partial L_{sm}}{\partial \boldsymbol{\lambda}}$ will have to include additional terms.

$$\frac{\partial \mathcal{T}}{\partial \boldsymbol{\lambda}} = \sum_{s=1}^{S} \sum_{j=1}^{N_s} \frac{\partial \mathcal{T}}{\partial \ln p_{sj}} \frac{\partial \ln p_{sj}}{\partial \boldsymbol{\lambda}} = \left(\frac{1}{\sum_{s=1}^{S} r_s} \right) \sum_{s=1}^{S} \sum_{j=1}^{N_s} p_{sj} \epsilon_{sj} \frac{\partial \ln p_{sj}}{\partial \boldsymbol{\lambda}} \tag{4}$$

$$\frac{\partial \ln p_{sj}}{\partial \boldsymbol{\lambda}} = \sum_{m=1}^{N_s} \frac{\partial \ln p_{sj}}{\partial L_{sm}} \frac{\partial L_{sm}}{\partial \boldsymbol{\lambda}} = \gamma \left(\frac{\partial L_{sj}}{\partial \boldsymbol{\lambda}} - \sum_{m=1}^{N_s} p_{sm} \frac{\partial L_{sm}}{\partial \boldsymbol{\lambda}} \right) \tag{5}$$

$$\frac{\partial L_{sm}}{\partial \boldsymbol{\lambda}} = \sum_{(s_k,t_k) \in h_{sm}} \frac{\partial L_{sm}}{\partial \ln P(s_k|t_k)} \frac{\partial \ln P(s_k|t_k)}{\partial \boldsymbol{\lambda}} = \sum_{(s_k,t_k) \in h_{sm}} \xi \frac{\partial \ln P(s_k|t_k)}{\partial \boldsymbol{\lambda}} \tag{6}$$

$$\frac{\partial \ln P(s_k|t_k)}{\partial \boldsymbol{\lambda}} = \sum_i \frac{\partial \ln P(s_k|t_k)}{\partial w_i} \frac{\partial w_i}{\partial \boldsymbol{\lambda}} \tag{7}$$

$$\frac{\partial \ln P(s_k|t_k)}{\partial w_i} = \frac{\sum_{j \in \mathcal{A}(s_k) \cap \mathcal{B}(t_k)} \delta\left(j-i\right) c_j(s_k,t_k)}{\sum_{j \in \mathcal{A}(s_k) \cap \mathcal{B}(t_k)} w_j c_j(s_k,t_k)} - \frac{\sum_{j \in \mathcal{B}(t_k)} \delta\left(j-i\right) c_j(t_k)}{\sum_{j \in \mathcal{B}(t_k)} w_j c_j(t_k)} \tag{8}$$

$$\delta(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0 \end{cases} \tag{9}$$

of LBFGS training, merging new hypotheses with translations from previous iterations. The overall training procedure is described in more detail below:

1. Initialize parameter vector $\boldsymbol{\lambda}$ to small random values, so that all training sentences receive approximately equal weights.
2. Initialize phrase-based MT decoding weights to previously tuned values.
3. Perform weighted phrase rule extraction as described in Equation 1, to estimate the phrase and lexical translation probabilities.
4. Decode the tuning set, generating N-best.
5. Merge N-best hypotheses from previous iterations to current N-best.
6. Tune decoding weights so as to minimize TER on merged N-best, using a derivative free optimization method. In our case, we used Powell's algorithm (Powell, 1964) modified by Brent as described in (Brent, 1973) [3].
7. Identify set of unique source-target phrase pairs in merged N-best.
8. Extract sufficient statistics from training data for all phrases identified in step 7.
9. Run the LBFGS algorithm to minimize the expected TER in the merged N-best, using the derivative equations described previously.
10. Assign a weight to each training sentence based on the $\boldsymbol{\lambda}$ values optimized in 9.
11. Go to step 3.

Typically, the corpus weights converge in about 4-5 main iterations. The calculation of the derivative is parallelized to speed up computation, requiring about 10 minutes per iteration of LBFGS.

4 Experimental Setup

In this section we describe the setup that was used for all experiments reported in this paper. Specifically, we provide details about the training data, development sets, and MT systems (phrase-based and hierarchical).

4.1 Training Data

All MT training experiments made use of an Arabic-English corpus of approximately 200 million tokens (English side). Most of the collections in this corpus are available through the Linguistic Data Consortium (LDC) and are regularly part of the resources specified for the constrained data track of the NIST MT evaluation[4].

[3] This method was first used for N-best based parameter optimization in (Ostendorf et al., 1991).

[4] For a list of the NIST MT09 constrained training condition resources, see `http://www.itl.nist.gov/iad/mig/tests/mt/2009/MT09_ConstrainedResources.pdf`

The corpus includes data from multiple genres, as shown in Table 1. The "Sakhr" newswire collection is a set of Arabic-to-English and English-to-Arabic data provided by Sakhr Software, totaling about 30.8 million tokens, and is only available to research teams participating in the Defense Advanced Research Projects Agency (DARPA) Global Autonomous Language Exploitation (GALE) program. The "LDC Gigaword (ISI)" collection was produced by automatically detecting and extracting portions of parallel text from the monolingual LDC Arabic and English Gigaword collections, using a method developed at the Information Sciences Institute (ISI) of the University of Southern California.

Data Origin	Style	Size (K tokens)
LDC pre-GALE	U. Nations	118049
	Newswire	2700
	Treebank	685
LDC post-GALE	Newswire	14344
	Treebank	292
	Web	478
	Broad. News	573
	Broad. Conv.	1003
Web-found text	Lexicons	436
	Quran	406
Sakhr	Newswire	30790
LDC Gigaword (ISI)	Newswire	29169

Table 1: Composition of the Arabic-English parallel corpus used for MT training.

It is easy to see that most of the parallel training data are either newswire or from United Nations. The amount of web text or broadcast news/conversations is only a very small fraction of the total corpus. In total, there are 31 collections in the training bitext. Some collections (especially those released recently by LDC for the GALE project) consist of data from multiple genres. The total number of unique genres (or data types) in the training set is 10.

Besides the above bitext, we also used approximately 8 billion words of English text for language model (LM) training (3.7B words from the LDC Gigaword corpus, 3.3B words of web-downloaded text, and 1.1B words of data from CNN archives). This data was used to train two language models: an entropy-pruned trigram LM, used in decoding, and an unpruned 5-gram LM used in N-best rescoring. Kneser-Ney smoothing was applied to the n-grams in both cases.

4.2 Development Sets

The development sets used for tuning and testing the corpus weights and other MT settings were comprised of documents from previous Arabic-English NIST MT evaluation sets and from GALE development/evaluation sets.

Specifically, the newswire Tune and Test sets consist of documents from the following collections: the newswire portion of NIST MT04, MT05, MT06, and MT08 evaluation sets, the GALE Phase 1 (P1) and Phase 2 (P2) evaluation sets, and the GALE P2 and P3 development sets. The web Tune and Test sets are made of documents from NIST MT06 and MT08, the GALE P1 and P2 evaluation sets, the GALE P2 and P3 development sets, and a held-out portion of the GALE year 1 quarter 4 web training data release.

The audio Tune and Test sets consist of roughly equal parts of news and conversations broadcast from November 2005 through May 2007 by major Arabic-speaking television and radio stations (e.g., Al-Jazeera, Al-Arabiya, Syrian TV), totaling approximately 14 hours of speech. The audio was processed through automated speech recognition (ASR) in order to produce (errorful) transcripts that were used as input to all MT decoding experiments reported in this paper. However, the corpus weight estimation was carried out based on N-best MT of the Arabic audio reference transcriptions (i.e., the transcripts had no speech recognition errors, and contained full punctuation).

It is important to note that some of the documents in the above devsets have multiple reference translations (usually 4), while others have only one. Most of the documents in the newswire sets have 4 references, but unfortunately the web and audio sets have, on average, less than 2 reference translations per segment. More details are listed in Table 2.

Another important note is that, although the audio sets consist of both broadcast news (BN) and broadcast conversations (BC), we did not perform BN or BC-specific tuning. Corpus weights and MT decoding parameters were optimized based on a single Tune set, on a mix of BN and BC data. However, when we report speech translation results in later sections, we break down the perfor-

Genre	Tune			Test		
	#segs	#tokens	#refs/seg	#segs	#tokens	#refs/seg
Newswire	1994	72359	3.94	3149	115700	3.67
Web	3278	99280	1.69	4425	125795	2.08
Audio BN	897	32990	1.00	1530	53067	1.00
Audio BC	765	24607	1.00	1416	44435	1.00

Table 2: Characteristics of the tuning (Tune) and validation (Test) sets used for development on Arabic newswire, web, and audio. The audio sets include material from both broadcast news and broadcast conversations.

mance by genre.

4.3 MT Systems

Experiments were performed using two types of statistical MT systems: a phrase-based system, similar to Pharaoh (Koehn, 2004), and a state-of-the-art, hierarchical string-to-dependency-tree system, similar to (Shen et al., 2008).

The phrase-based MT system employs a pruned 3-gram LM in decoding, and can optionally generate N-best unique translation hypotheses which are used to estimate the corpus weights, as described in Section 3.

The hierarchical MT system performs decoding with the same 3-gram LM, generates N-best of unique translation hypotheses, and then rescores them using a large, unpruned 5-gram LM in order to select the best scoring translation. It is worth mentioning that this hierarchical MT system provides a very strong baseline; it achieves a case-sensitive BLEU score of 52.20 on the newswire portion of the NIST MT08 evaluation set, which is similar to the score of the second-best system that participated in the unconstrained data track of the NIST MT08 evaluation.

Both types of models were trained on the same word alignments generated by GIZA++ (Och and Ney, 2003).

5 Results

In this section we report results on the Arabic newswire, web, and audio development sets, using both phrase-based and hierarchical MT systems, in terms of TER, BLEU[5], and METEOR (Lavie and Agarwal, 2007). Whenever corpus weights are used, they were estimated on the designated Tune set using the phrase-based MT system. Only the collection and genre ids were used as sentence features in order to estimate the corpus weights. As mentioned in Section 4.1, the training bitext consists of 31 collections and 10 genres, so each training sentence was assigned a 41-dimensional binary vector indicating its particular collection/genre combination. That vector was then mapped into a single weight using a perceptron.

5.1 Phrase-based MT

Results using the phrase-based MT system are shown in Table 3. In all cases, the decoding weights were optimized so as to minimize TER on the designated Tune set. On newswire, the discriminative corpus weights provide 0.8% absolute gain in TER, in both Tune and Test sets. On web, the TER gain is 0.9% absolute on Tune and 0.5% on Test. On the audio Test set, the TER gain is 0.5% on BN and 1.4% on BC. Significant improvements were also obtained in the BLEU and METEOR scores, on all sets and conditions.

5.2 Hierarchical MT

Results using the hierarchical MT system are shown in Table 4. The hierarchical system used different tuning criteria in each genre. On newswire, the decoding weights were optimized so as to maximize BLEU, while on web and audio the tuning was based on $0.5\text{TER} + 0.5(1 - \text{BLEU})$ (referred to as TERBLEU in what follows). Note that these were the criteria for tuning the decoding weights; whenever corpus weights were used, they were taken from the phrase-based system.

It is interesting to see that gains from discriminative corpus weights carry over to the more powerful hierarchical MT system. On newswire Test, the gain in BLEU is 0.8; on web Test, the gain in TERBLEU is 0.3. On the audio Test set, the corpus weights provide 0.7 and 0.75 TERBLEU reduction on BN and BC, respectively. As with the

[5]The brevity penalty was calculated using the formula in the original IBM paper, rather than the more recent definition implemented in the NIST mteval-v11b.pl script.

Set	Corpus Weights	Newswire			Web		
		TER	BLEU	MTR	TER	BLEU	MTR
Tune	No	42.3	48.2	67.5	60.0	21.9	51.3
	Yes	41.5	49.6	68.7	59.1	22.8	52.3
Test	No	43.2	46.2	66.5	58.6	24.2	52.2
	Yes	42.4	47.5	67.8	58.1	25.4	52.9

(a) Results on Arabic text.

Set	Corpus Weights	BN			BC		
		TER	BLEU	MTR	TER	BLEU	MTR
Tune	No	56.0	22.9	55.5	57.3	21.7	55.0
	Yes	55.0	25.0	57.1	56.1	23.6	56.4
Test	No	53.0	25.3	57.7	55.9	22.9	55.4
	Yes	52.5	26.6	58.8	54.5	24.7	56.8

(b) Results on Arabic audio.

Table 3: Phrase-based trigram decoding results on the Arabic text and audio development sets. Decoding weights were optimized on the Tune set in order to directly minimize TER. Corpus weights were also optimized on Tune set, but based on expected TER.

phrase-based system, all metrics improve from the use of corpus weights, in all sets/conditions.

6 Conclusions

We have described a novel approach for estimating a weight for each sentence in a parallel training corpus so as to optimize MT performance of a phrase-based statistical MT system. The sentence weights influence MT performance by being applied to the phrase and lexical counts during translation rule extraction and probability estimation.

In order to ensure robust training of the weights, we expressed them as a function of sentence-level features. Then, we defined the process for optimizing the parameters of that function based on the expected TER of a translation hypothesis N-best on a designated tuning set.

The proposed technique was evaluated in the context of Arabic-English translation, on multiple conditions. It was shown that encouraging results were obtained by just using collection and genre ids as features. Interestingly, the discriminative corpus weights were found to be generally applicable and provided gains in a state-of-the-art hierarchical string-to-dependency-tree MT system, even though they were trained using the phrase-based MT system.

Next step is to include other sentence-level features, as described in Section 3.1. Finally, the technique described in this paper can be extended to address the estimation of weights at the alignment link level, based on link-level features. We believe that this will have a larger impact on the lexical and phrase translation probabilities, since there is a large number of parallel training sentences that are partially correct, i.e., they contain parts that are aligned and translated correctly, and parts that are wrong. The current procedure tries to assign a single weight to such sentences, so there is no way to distinguish between the "good" and "bad" portions of each sentence. Pushing the weight estimation at the alignment link level will alleviate this problem and will make the discriminative training more targeted.

Acknowledgments

This work was supported by DARPA/IPTO Contract No. HR0011-06-C-0022 under the GALE program.

References

Richard P. Brent. 1973. *Algorithms for Minimization Without Derivatives*. Prentice-Hall.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the transla-

Set	Corpus Weights	Newswire			Web		
		TER	BLEU	MTR	TER	BLEU	MTR
Tune	No	39.5	54.4	70.3	58.2	25.2	53.8
	Yes	38.8	55.6	71.2	58.0	25.5	54.0
Test	No	40.7	52.1	69.3	57.0	28.3	54.7
	Yes	40.1	52.9	69.8	56.6	28.5	55.0

(a) Results on Arabic text.

Set	Corpus Weights	BN			BC		
		TER	BLEU	MTR	TER	BLEU	MTR
Tune	No	54.9	27.3	58.0	55.8	26.1	57.4
	Yes	53.6	28.2	59.0	54.9	26.9	58.0
Test	No	51.6	29.9	60.0	54.4	27.6	57.7
	Yes	50.7	30.4	60.7	53.2	27.9	58.7

(b) Results on Arabic audio.

Table 4: Hierarchical 5-gram rescoring results on the Arabic text and audio development sets. Decoding/rescoring weights were optimized on the Tune set in order to directly maximize BLEU (for newswire) or minimize TERBLEU (for web and audio). Corpus weights were the same as the ones used in the corresponding phrase-based decodings.

tion model for statistical machine translation based on information retrieval. In *Proceedings of the 10th Annual Conference of European Association for Machine Translation*, pages 133–142.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas*.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528.

Yajuan Lu, Jin Huang, and Qun Liu. 2007. Improving statistical machine translation performance by training data selection and optimization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 343–350.

Arindam Mandal, Dimitra Vergyri, Wen Wang, Jing Zheng, Andreas Stolcke, Gokhan Tur, Dilek Hakkani-Tür, and Necip Fazil Ayan. 2008. Efficient data selection for machine translation. In *Proceedings of the Second IEEE/ACL Spoken Language Technology Workshop*, pages 261–264.

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

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

M. Ostendorf, A. Kannan, S. Austin, O. Kimball, R. Schwartz, and J. R. Rohlicek. 1991. Integration of diverse recognition methodologies through reevaluation of nbest sentence hypotheses. In *Proceedings of the DARPA Workshop on Speech and Natural Language*, pages 83–87.

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

M. J. D. Powell. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, pages 155–162.

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

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 577–585.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciula, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231.

Matthew Snover, Bonnie Dorr, and Richard Schwartz. 2008. Language and translation model adaptation using comparable corpora. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 857–866.

Keiji Yasuda, Ruiqiang Zhang, Hirofumi Yamamoto, and Eiichiro Sumita. 2008. Method of selecting training data to build a compact and efficient translation model. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, volume II, pages 655–660.

Unsupervised Tokenization for Machine Translation

Tagyoung Chung and **Daniel Gildea**
Computer Science Department
University of Rochester
Rochester, NY 14627

Abstract

Training a statistical machine translation starts with tokenizing a parallel corpus. Some languages such as Chinese do not incorporate spacing in their writing system, which creates a challenge for tokenization. Moreover, morphologically rich languages such as Korean present an even bigger challenge, since optimal token boundaries for machine translation in these languages are often unclear. Both rule-based solutions and statistical solutions are currently used. In this paper, we present unsupervised methods to solve tokenization problem. Our methods incorporate information available from parallel corpus to determine a good tokenization for machine translation.

1 Introduction

Tokenizing a parallel corpus is usually the first step of training a statistical machine translation system. With languages such as Chinese, which has no spaces in its writing system, the main challenge is to segment sentences into appropriate tokens. With languages such as Korean and Hungarian, although the writing systems of both languages incorporate spaces between "words", the granularity is too coarse compared with languages such as English. A single word in these languages is composed of several morphemes, which often correspond to separate words in English. These languages also form compound nouns more freely. Ideally, we want to find segmentations for source and target languages that create a one-to-one mapping of words. However, this is not always straightforward for two major reasons. First, what the optimal tokenization for machine translation should be is not always clear. Zhang et al. (2008b) and Chang et al. (2008) show that getting the tokenization of one of the languages in the corpus close to a gold standard does not necessarily help with building better machine translation systems. Second, even statistical methods require hand-annotated training data, which means that in resource-poor languages, good tokenization is hard to achieve.

In this paper, we explore unsupervised methods for tokenization, with the goal of automatically finding an appropriate tokenization for machine translation. We compare methods that have access to parallel corpora to methods that are trained solely using data from the source language. Unsupervised monolingual segmentation has been studied as a model of language acquisition (Goldwater et al., 2006), and as model of learning morphology in European languages (Goldsmith, 2001). Unsupervised segmentation using bilingual data has been attempted for finding new translation pairs (Kikui and Yamamoto, 2002), and for finding good segmentation for Chinese in machine translation using Gibbs sampling (Xu et al., 2008). In this paper, further investigate the use of bilingual information to find tokenizations tailored for machine translation. We find a benefit not only for segmentation of languages with no space in the writing system (such as Chinese), but also for the smaller-scale tokenization problem of normalizing between languages that include more or less information in a "word" as defined by the writing system, using Korean-English for our experiments. Here too, we find a benefit from using bilingual information, with unsupervised segmentation rivaling and in some cases surpassing supervised segmentation. On the modeling side, we use dynamic programming-based variational Bayes, making Gibbs sampling unnecessary. We also develop and compare various factors in the model to control the length of the tokens learned, and find a benefit from adjusting these parameters directly to optimize the end-to-end translation quality.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 718–726,
Singapore, 6-7 August 2009.

2 Tokenization

Tokenization is breaking down text into lexemes — a unit of morphological analysis. For relatively isolating languages such as English and Chinese, a word generally equals a single token, which is usually a clearly identifiable unit. English, especially, incorporates spaces between words in its writing system, which makes tokenization in English usually trivial. The Chinese writing system does not have spaces between words, but there is less ambiguity where word boundaries lie in a given sentence compared to more agglutinative languages. In languages such as Hungarian, Japanese, and Korean, what constitutes an optimal token boundary is more ambiguous. While two tokens are usually considered two separate words in English, this may be not be the case in agglutinative languages. Although what is considered a single morphological unit is different from language to language, if someone were given a task to align words between two languages, it is desirable to have one-to-one token mapping between two languages in order to have the optimal problem space. For machine translation, one token should not necessarily correspond to one morphological unit, but rather should reflect the morphological units and writing system of the other language involved in translation.

For example, consider a Korean "word" *meok-eoss-da*, which means *ate*. It is written as a single word in Korean but consists of three morphemes *eat-past-indicative*. If one uses morphological analysis as the basis for Korean tokenization, *meok-eoss-da* would be split into three tokens, which is not desirable if we are translating Korean to English, since English does not have these morphological counterparts. However, a Hungarian word *szekrényemben*, which means *in my closet*, consists of three morphemes *closet-my-inessive* that are distinct words in English. In this case, we do want our tokenizer to split this "word" into three morphemes *szekrény em ben*.

In this paper, we use segmentation and tokenization interchangeably as blanket terms to cover the two different problems we have presented here. The problem of segmenting Chinese sentences into words and the problem of segmenting Korean or Hungarian "words" into tokens of right granularity are different in their nature. However, our models presented in section 3 handle the both problems.

3 Models

We present two different methods for unsupervised tokenization. Both are essentially unigram tokenization models. In the first method, we try learning tokenization from word alignments with a model that bears resemblance to Hidden Markov models. We use IBM Model 1 (Brown et al., 1993) for the word alignment model. The second model is a relatively simpler monolingual tokenization model based on counts of substrings which serves as a baseline of unsupervised tokenization.

3.1 Learning tokenization from alignment

We use expectation maximization as our primary tools in learning tokenization form parallel text. Here, the observed data provided to the algorithm are the tokenized English string $\mathbf{e}_1^n$ and the untokenized string of foreign characters $\mathbf{c}_1^m$. The unobserved variables are both the word-level alignments between the two strings, and the tokenization of the foreign string. We represent the tokenization with a string $\mathbf{s}_1^m$ of binary variables, with $s_i = 1$ indicating that the ith character is the final character in a word. The string of foreign words $\mathbf{f}_1^\ell$ can be thought of as the result of applying the tokenization $\mathbf{s}$ to the character string $\mathbf{c}$:

$$\mathbf{f} = \mathbf{s} \circ \mathbf{c} \quad \text{where} \quad \ell = \sum_{i=1}^{m} s_i$$

We use IBM Model 1 as our word-level alignment model, following its assumptions that each foreign word is generated independently from one English word:

$$\begin{aligned} P(\mathbf{f}|\mathbf{e}) &= \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) \\ &= \sum_{\mathbf{a}} \prod_i P(f_i \mid e_{a_i}) P(\mathbf{a}) \\ &= \prod_i \sum_j P(f_i \mid e_j) P(a_i = j) \end{aligned}$$

and that all word-level alignments $\mathbf{a}$ are equally likely: $P(a) = \frac{1}{n}$ for all positions. While Model 1 has a simple EM update rule to compute posteriors for the alignment variables $\mathbf{a}$ and from them learn the lexical translation parameters $P(f \mid e)$, we cannot apply it directly here because $\mathbf{f}$ itself is unknown, and ranges over an exponential number of possibilities depending on the hidden segmentation $\mathbf{s}$. This can be addressed by applying dynamic programing over the sequence $\mathbf{s}$. We compute the

posterior probability of a word beginning at position i, ending at position j, and being generated by English word k:

$$P(\mathbf{s}_{i...j} = (1, 0, \ldots, 0, 1), a = k \mid \mathbf{e}) = \frac{\alpha(i)P(f \mid e_k)P(a = k)\beta(j)}{P(\mathbf{c} \mid \mathbf{e})}$$

where $f = c_i \ldots c_j$ is the word formed by concatenating characters i through j, and a is a variable indicating which English position generated f. Here α and β are defined as:

$$\alpha(i) = P(\mathbf{c}_1^i, s_i = 1 \mid \mathbf{e})$$
$$\beta(j) = P(\mathbf{c}_{j+1}^m, s_j = 1 \mid \mathbf{e})$$

These quantities resemble forward and backward probabilities of hidden Markov models, and can be computed with similar dynamic programming recursions:

$$\alpha(i) = \sum_{\ell=1}^{L} \alpha(i - \ell) \sum_a P(a)P(c_{i-\ell}^i \mid e_a)$$
$$\beta(j) = \sum_{\ell=1}^{L} \sum_a P(a)P(c_j^{j+\ell} \mid e_a)\beta(j + \ell)$$

where L is the maximum character length for a word.

Then, we can calculate the expected counts of individual word pairs being aligned (c_i^j, e_k) by accumulating these posteriors over the data:

$$ec(c_i^j, e_k) \mathrel{+}= \frac{\alpha(i)P(a)P(c_i^j \mid e_k)\beta(j)}{\alpha(m)}$$

The M step simply normalizes the counts:

$$\tilde{P}(f \mid e) = \frac{ec(f, e)}{\sum_e ec(f, e)}$$

Our model can be compared to a hidden Markov model in the following way: a target word generates a source token which spans a zeroth order Markov chain of characters in source sentence, where a "transition" represents a segmentation and a "emission" represents an alignment. The model uses HMM-like dynamic programming to do inference. For the convenience, we refer to this model as the bilingual model in the rest of the paper. Figure 1 illustrates our first model with an small example. Under this model we are not learning segmentation directly, but rather we are learning alignments between two sentences. The segmentation is by-product of learning the alignment. We can find the optimal segmentation of a new source language sentence using the Viterbi algorithm. Given two sentences $\mathbf{e}$ and $\mathbf{f}$,

$$\mathbf{a}^* = \operatorname*{argmax}_{\mathbf{a}} P(\mathbf{f}, \mathbf{a} \mid \mathbf{e})$$

and segmentation $\mathbf{s}^*$ implied by alignment $\mathbf{a}^*$ is the optimal segmentation of $\mathbf{f}$ found by this model.

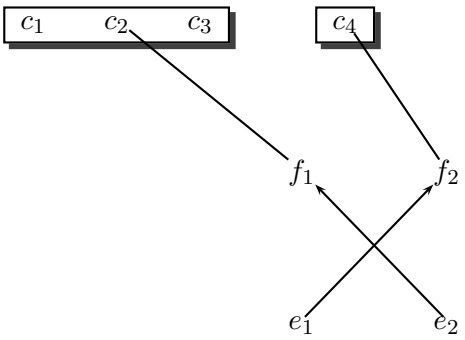


Figure 1: The figure shows a source sentence $\mathbf{f} = f_1, f_2 = \mathbf{s} \circ c_1 \ldots c_4$ where $\mathbf{s} = (0, 0, 1, 1)$ and a target sentence $\mathbf{e} = e_1, e_2$. There is a segmentation between c_3 and c_4; thus c_1, c_2, c_3 form f_1 and c_3 forms f_2. f_1 is generated by e_2 and f_2 is generated by e_1.

3.2 Learning tokenization from substring counts

The second tokenization model we propose is much simpler. More sophisticated unsupervised monolingual tokenization models using hierarchical Bayesian models (Goldwater et al., 2006) and using the minimum description length principle (Goldsmith, 2001; de Marcken, 1996) have been studied. Our model is meant to serve as a computationally efficient baseline for unsupervised monolingual tokenization. Given a corpus of only source language of unknown tokenization, we want to find the optimal $\mathbf{s}$ given $\mathbf{c}$ — $\mathbf{s}$ that gives us the highest $P(\mathbf{s} \mid \mathbf{c})$. According to Bayes' rule,

$$P(\mathbf{s} \mid \mathbf{c}) \propto P(\mathbf{c} \mid \mathbf{s})P(\mathbf{s})$$

Again, we assume that all $P(\mathbf{s})$ are equally likely. Let $\mathbf{f} = \mathbf{s} \circ \mathbf{c} = f_1 \ldots f_\ell$, where f_i is a word under some possible segmentation $\mathbf{s}$. We want to find the $\mathbf{s}$ that maximizes $P(\mathbf{f})$. We assume that

$$P(\mathbf{f}) = P(f_1) \times \ldots \times P(f_\ell)$$

To calculate $P(f_i)$, we count every possible

substring — every possible segmentation of characters — from the sentences. We assume that

$$P(f_i) = \frac{count(f_i)}{\sum_k count(f_k)}$$

We can compute these counts by making a single pass through the corpus. As in the bilingual model, we limit the maximum size of f for practical reasons and to prevent our model from learning unnecessarily long f. With $P(f)$, given a sequence of characters $\mathbf{c}$, we can calculate the most likely segmentation using the Viterbi algorithm.

$$\mathbf{s}^* = \underset{\mathbf{s}}{\operatorname{argmax}}\, P(\mathbf{f})$$

Our rationale for this model is that if a span of characters $f = c_i \dots c_j$ is an independent token, it will occur often enough in different contexts that such a span of characters will have higher probability than other spans of characters that are not meaningful. For the rest of the paper, this model will be referred to as the monolingual model.

3.3 Tokenizing new data

Since the monolingual tokenization only uses information from a monolingual corpus, tokenizing new data is not a problem. However, with the bilingual model, we are learning $P(f \mid e)$. We are relying on information available from **e** to get the best tokenization for **f**. However, the parallel sentences will not be available for new data we want to translate. Therefore, for the new data, we have to rely only on $P(f)$ to tokenize any new data, which can be obtained by calculating

$$P(f) = \sum_e P(f \mid e)P(e)$$

With $P(f)$ from the bilingual model, we can run the Viterbi algorithm in the same manner as monolingual tokenization model for monolingual data. We hypothesize that we can learn valuable information on which token boundaries are preferable in language f when creating a statistical machine translation system that translates from language f to language e.

4 Preventing overfitting

We introduce two more refinements to our word-alignment induced tokenization model and monolingual tokenization model. Since we are considering every possible token f that can be guessed from our corpus, the data is very sparse. For the bilingual model, we are also using the EM algorithm to learn $P(f \mid e)$, which means there is a danger of the EM algorithm memorizing the training data and thereby overfitting. We put a Dirichlet prior on our multinomial parameter for $P(f \mid e)$ to control this situation. For both models, we also want a way to control the distribution of token length after tokenization. We address this problem by adding a length factor to our models.

4.1 Variational Bayes

Beal (2003) and Johnson (2007) describe variational Bayes for hidden Markov model in detail, which can be directly applied to our bilingual model. With this Bayesian extension, the emission probability of our first model can be summarized as follows:

$$\theta_{\mathbf{e}} \mid \alpha \sim \mathrm{Dir}(\alpha),$$
$$f_i \mid e_i = e \sim \mathrm{Multi}(\theta_{\mathbf{e}}).$$

Johnson (2007) and Zhang et al. (2008a) show having small α helps to control overfitting. Following this, we set our Dirichlet prior to be as sparse as possible. It is set at $\alpha = 10^{-6}$, the number we used as floor of our probability.

For the model incorporating the length factor, which is described in the next section, we do not place a prior on our transition probability, since there are only two possible states, i.e. $P(s = 1)$ and $P(s = 0)$. This distribution is not as sparse as the emission probability.

Comparing variational Bayes to the traditional EM algorithm, the E step stays the same but the M step for calculating the emission probability changes as follows:

$$\tilde{P}(f \mid e) = \frac{\exp(\psi(ec(f, e) + \alpha))}{\exp(\psi(\sum_e ec(f, e) + s\alpha))}$$

where ψ is the digamma function, and s is the size of the vocabulary from which f is drawn. Since we do not accurately know s, we set s to be the number of all possible tokens. As can be seen from the equation, by setting α to a small value, we are discounting the expected count with help of the digamma function. Thus, having lower α leads to a sparser solution.

4.2 Token length

We now add a parameter that can adjust the tokenizer's preference for longer or shorter tokens.

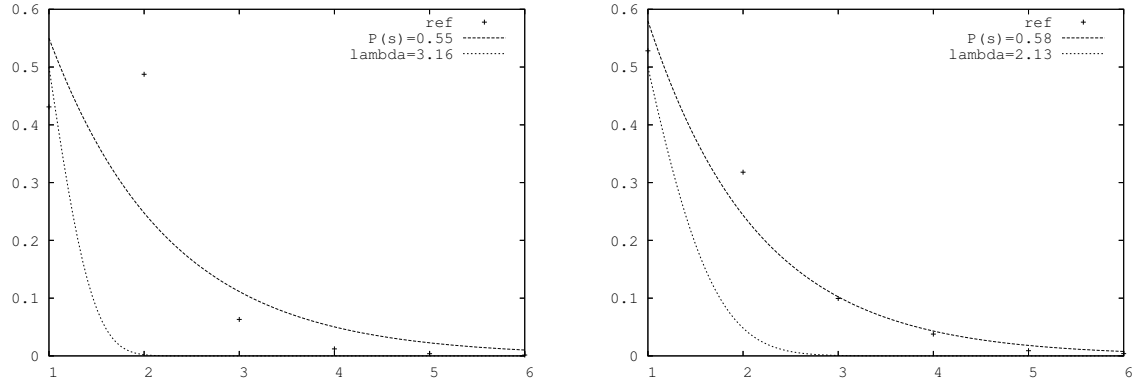


Figure 2: Distribution of token length for (from left to right) Chinese, and Korean. "ref" is the empirical distribution from supervised tokenization. Two length factors — ϕ_1 and ϕ_2 are also shown. For ϕ_1, the parameter to geometric distribution $P(s)$ is set to the value learned from our bilingual model. For ϕ_2, λ is set using the criterion described in the experiment section.

This parameter is beneficial because we want our distribution of token length after tokenization to resemble the real distribution of token length. This parameter is also useful because we also want to incorporate information on the number of tokens in the other language in the parallel corpus. This is based on the assumption that, if tokenization creates a one-to-one mapping, the number of tokens in both languages should be roughly the same. We can force the two languages to have about the same number of tokens by adjusting this parameter. The third reason is to further control overfitting. Our observation is that certain morphemes are very common, such that they will be always observed attached to other morphemes. For example, in Korean, a noun attached with nominative case marker is very common. Our model is likely to learn a noun attached with the morpheme — nominative case marker — rather than noun itself. This is not desirable when the noun occurs with less common morphemes; in these cases the morpheme will be split off creating inconsistencies.

We have experimented with two different length factors, each with one adjustable parameter:

$$\phi_1(\ell) = P(s)(1 - P(s))^{\ell-1}$$
$$\phi_2(\ell) = 2^{-\ell^\lambda}$$

The first, ϕ_1, is the geometric distribution, where l is length of a token and $P(s)$ is probability of segmentation between two characters. The second length factor ϕ_2 was acquired through several experiments and was found to work well. As can been seen from Figure 2, the second factor discounts longer tokens more heavily than the geometric distribution. We can adjust the value of λ and $P(s)$ to increase or decrease number of tokens after segmentation.

For our monolingual model, incorporating these factors is straightforward. We assume that

$$P(\mathbf{f}) \propto P(f_1)\phi(\ell_1) \times \ldots \times P(f_n)\phi(\ell_n)$$

where ℓ_i is the length of f_i. Then, we use the same Viterbi algorithm to select the $f_1 \ldots f_n$ that maximizes $P(\mathbf{f})$, thereby selecting the optimal $\mathbf{s}$ according to our monolingual model with a length factor. We pick the value of λ and $P(s)$ that produces about the same number of tokens in the source side as in the target side, thereby incorporating some information about the target language.

For our bilingual model, we modify our model slightly to incorporate ϕ_1, creating a hybrid model. Now, our forward probability of forward-backward algorithm is:

$$\alpha(i) = \sum_{\ell=1}^{L} \alpha(i-l)\phi_1(\ell) \sum_a P(a)P(c_{i-\ell}^{i} \mid e_a)$$

and the expected count of (c_i^j, e_k) is

$$ec(c_i^j, e_k) \mathrel{+}= \frac{\alpha(i)P(a)P(c_i^j \mid e_k)\beta(j)\phi_1(j-i)}{\alpha(m)}$$

For ϕ_1, we can learn $P(s)$ for the geometric distribution from the model itself:[1]

$$P(s) = \frac{1}{m}\sum_{i}^{m} \frac{\alpha(i)\beta(i)}{\alpha(m)}$$

[1]The equation is for one sentence, but in practice, we sum over all sentences in the training data to calculate $P(s)$.

We can also fix $P(s)$ instead of learning it through EM. We incorporate ϕ_2 into the bilingual model as follows: after learning $P(f)$ from the bilingual model, we pick the λ in the same manner as the monolingual model and run the Viterbi algorithm.

After applying the length factor, what we have is a log-linear model for tokenization, with two feature functions with equal weights: the length factor and $P(f)$ learned from model.

5 Experiments

5.1 Data

We tested our tokenization methods on two different language pairs: Chinese-English, and Korean-English. For Chinese-English, we used FBIS newswire data. The Korean-English parallel data was collected from news websites and sentence-aligned using two different tools described by Moore (2002) and Melamed (1999). We used subsets of each parallel corpus consisting of about 2M words and 60K sentences on the English side. For our development set and test set, Chinese-English had about 1000 sentences each with 10 reference translations taken from the NIST 2002 MT evaluation. For Korean-English, 2200 sentence pairs were randomly sampled from the parallel corpus, and held out from the training data. These were divided in half and used for test set and development set respectively. For all language pairs, very minimal tokenization — splitting off punctuation — was done on the English side.

5.2 Experimental setup

We used Moses (Koehn et al., 2007) to train machine translation systems. Default parameters were used for all experiments except for the number of iterations for GIZA++ (Och and Ney, 2003). GIZA++ was run until the perplexity on development set stopped decreasing. For practical reasons, the maximum size of a token was set at three for Chinese, and four for Korean.[2] Minimum error rate training (Och, 2003) was run on each system afterwards and BLEU score (Papineni et al., 2002) was calculated on the test sets.

For the monolingual model, we tested two versions with the length factor ϕ_1, and ϕ_2. We picked λ and $P(s)$ so that the number of tokens on source side (Chinese, and Korean) will be about the same as the number of tokens in the target side (English).

For the bilingual model, as explained in the model section, we are learning $P(f \mid e)$, but only $P(f)$ is available for tokenizing any new data. We compared two conditions: using only the source data to tokenize the source language training data according to $P(f)$ (which is consistent with the conditions at test time), and using both the source and English data to tokenize the source language training data (which might produce better tokenization by using more information). For the first length factor ϕ_1, we ran an experiment where the model learns $P(s)$ as described in the model section, and we also had experiments where $P(s)$ was pre-set at 0.9, 0.7, 0.5, and 0.3 for comparison. We also ran an experiment with the second length factor ϕ_2 where λ was picked as the same manner as the monolingual model.

We varied tokenization of development set and test set to match the training data for each experiment. However, as we have implied in the previous paragraph, in the one experiment where $P(f \mid e)$ was used to segment training data, directly incorporating information from target corpus, tokenization for test and development set is not exactly consistent with tokenization of training corpus. Since we assume only source corpus is available at the test time, the test and the development set was tokenized only using information from $P(f)$.

We also trained MT systems using supervised tokenizations and tokenization requiring a minimal effort for the each language pair. For Chinese-English, the minimal effort tokenization is maximal tokenization where every Chinese character is segmented. Since a number of Chinese tokenizers are available, we have tried four different tokenizations for the supervised tokenizations. The first one is the LDC Chinese tokenizer available at the LDC website[3], which is compiled by Zhibiao Wu. The second tokenizer is a maxent-based tokenizer described by Xue (2003). The third and fourth tokenizations come from the CRF-based Stanford Chinese segmenter described by Chang et al. (2008). The difference between third and fourth tokenization comes from the different gold standard, the third one is based on Beijing University's segmentation (pku) and the fourth one is based on Chinese Treebank (ctb). For Korean-

[2] In the Korean writing system, one character is actually one syllable block. We do not decompose syllable blocks into individual consonants and vowels.

[3] http://projects.ldc.upenn.edu/Chinese/LDC_ch.htm

	Chinese		Korean
	BLEU	F-score	BLEU
Supervised			
Rule-based morphological analyzer			7.27
LDC segmenter	20.03	0.94	
Xue's segmenter	23.02	0.96	
Stanford segmenter (pku)	21.69	0.96	
Stanford segmenter (ctb)	22.45	1.00	
Unsupervised			
Splitting punctuation only			6.04
Maximal (Character-based MT)	20.32	0.75	
Bilingual $P(f \mid e)$ with ϕ_1 $P(s) = learned$	19.25		6.93
Bilingual $P(f)$ with ϕ_1 $P(s) = learned$	20.04	0.80	7.06
Bilingual $P(f)$ with ϕ_1 $P(s) = 0.9$	20.75	0.87	**7.46**
Bilingual $P(f)$ with ϕ_1 $P(s) = 0.7$	20.59	0.81	7.31
Bilingual $P(f)$ with ϕ_1 $P(s) = 0.5$	19.68	0.80	7.18
Bilingual $P(f)$ with ϕ_1 $P(s) = 0.3$	20.02	0.79	7.38
Bilingual $P(f)$ with ϕ_2	**22.31**	0.88	7.35
Monolingual $P(f)$ with ϕ_1	20.93	0.83	6.76
Monolingual $P(f)$ with ϕ_2	20.72	0.85	7.02

Table 1: BLEU score results for Chinese-English and Korean-English experiments and F-score of segmentation compared against Chinese Treebank standard. The highest unsupervised score is highlighted.

English, the minimal effort tokenization splitting off punctuation and otherwise respecting the spacing in the Korean writing system. A Korean morphological analysis tool[4] was used to create the supervised tokenization.

For Chinese-English, since a gold standard for Chinese segmentation is available, we ran an additional evaluation of tokenization from each methods we have tested. We tokenized the raw text of Chinese Treebank (Xia et al., 2000) using all of the methods (supervised/unsupervised) we have described in this section except for the bilingual tokenization using $P(f \mid e)$ because the English translation of the Chinese Treebank data was not available. We compared the result against the gold standard segmentation and calculated the F-score.

6 Results

Results from Chinese-English and Korean-English experiments are presented in Table 1. Note that nature of data and number of references are different for the two language pairs, and therefore the BLEU scores are not comparable. For both language pairs, our models perform equally well as supervised baselines, or even better. We can observe three things from the result. First, tokenization of training data using $P(f \mid e)$ tested on a test set tokenized with $P(f)$ performed worse than any other experiments. This affirms our belief that consistency in tokenization is important for machine translation, which was also mentioned by Chang et al. (2008). Secondly, we are learning valuable information by looking at the target language. Compare the result of the bilingual model with ϕ_2 as the length factor to the result of the monolingual model with the same length factor. The bilingual version consistently performed better than the monolingual model in all language pairs. This tells us we can learn better token boundaries by using information from the target language. Thirdly, our hypothesis on the need for heavy discount for longer tokens is confirmed. The value for $P(s)$ learned by the model was 0.55, and 0.58 for Chinese, and Korean respectively. For both language pairs, this accurately reflects the empirical distribution of token length, as can be seen in Figure 2. However, experiments where $P(s)$ was directly optimized performed better, indicating that this parameter should be optimized within the context of a complete system. The second length factor ϕ_2, which discounts longer tokens even more heavily, generally performed bet-

[4] http://nlp.kookmin.ac.kr/HAM/eng/main-e.html

English	the two presidents will hold a joint press conference at the end of their summit talks .
Untokenized Korean	양국 정상은 회담이 끝난 뒤 공동 기자회견을 갖고 회담 결과를 공식 발표한다 .
Supervised	양국 정상␣은 회담␣이 끝나␣ㄴ 뒤 공동 기자회견␣을 갖␣고 회담 결과␣를 공식 발표␣하␣ㄴ다 .
Bilingual $P(f \mid e)$ with ϕ_1	양국 정상은 회담␣이 끝난 뒤 공동 기자회견␣을 갖고 회담 결과␣를 공식 발표한␣다 .
Bilingual $P(f)$ with ϕ_2	양국 정상␣은 회담␣이 끝난 뒤 공동 기자회견␣을 갖고 회담 결과␣를 공식 발표␣한다 .
Monolingual $P(f)$ with ϕ_1	양국 정␣상␣은 회담␣이 끝난 뒤 공동 기자회견␣을 갖고 회담 결과를 공식 발표한␣다 .
Monolingual $P(f)$ with ϕ_2	양국 정상␣은 회담␣이 끝난 뒤 공동 기자회견␣을 갖고 회담 결과를 공식 발표␣한다 .

Figure 3: Sample tokenization results for Korean-English data. The underscores are added to clearly visualize where the breaks are.

ter than the first length factor when used in conjunction with the bilingual model. Lastly, F-scores of Chinese segmentations compared against the gold standard shows higher segmentation accuracy does not necessarily lead to higher BLEU score. F-scores presented in Table 1 are not directly comparable for all different experiments because the test data (Chinese Treebank) is used in training for some of the supervised segmenters, but these numbers do show how close unsupervised segmentations are to the gold standard. It is interesting to note that our highest unsupervised segmentation result does make use of bilingual information.

Sample tokenization results for Korean-English experiments are presented in Figure 3. We observe that different configurations produce different tokenizations, and the bilingual model produced generally better tokenizations for translation compared to the monolingual models or the supervised tokenizer. In this example, the tokenization obtained from the supervised tokenizer, although morphologically correct, is too fine-grained for the purpose of translation to English. For example, it correctly tokenized the attributive suffix ㄴ *-n* however, this is not desirable since English has no such counterpart. Both variations of the monolingual tokenization have errors such as incorrectly not segmenting 결과를 *gyeol-gwa-reul*, which is a compound of a noun and a case marker, into 결과␣를 *gyeol-gwa␣reul* as the bilingual model was able to do.

6.1 Conclusion and future work

We have shown that unsupervised tokenization for machine translation is feasible and can outperform rule-based methods that rely on lexical analysis, or supervised statistical segmentations. The approach can be applied both to morphological analysis of Korean and the segmentation of sentences into words for Chinese, which may at first glace appear to be quite different problems. We have only shown how our methods can be applied to one language of the pair, where one language is generally isolating and the other is generally synthetic. However, our methods could be extended to tokenization for both languages by iterating between languages. We also used the most simple word-alignment model, but more complex word alignment models could be incorporated into our bilingual model.

Acknowledgments This work was supported by NSF grants IIS-0546554 and ITR-0428020.

References

Matthew J. Beal. 2003. *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, University College London.

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

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

Carl de Marcken. 1996. Linguistic structure as composition and perturbation. In *Meeting of the Association for Computational Linguistics*, pages 335–341. Morgan Kaufmann Publishers.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the International Conference on Computational Linguistics/Association for Computational Linguistics (COLING/ACL-06)*, pages 673–680.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *2007 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 296–305, Prague, Czech Republic, June. Association for Computational Linguistics.

Genichiro Kikui and Hirofumi Yamamoto. 2002. Finding translation pairs from english-japanese untokenized aligned corpora. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02) workshop on Speech-to-speech translation: algorithms and systems*, pages 23–30. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Conference of the Association for Computational Linguistics (ACL-07), Demonstration Session*, pages 177–180.

I. Dan Melamed. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25:107–130.

Robert C. Moore. 2002. Fast and accurate sentence alignment of bilingual corpora. In *AMTA '02: Proceedings of the 5th Conference of the Association for Machine Translation in the Americas on Machine Translation: From Research to Real Users*, pages 135–144, London, UK. Springer-Verlag.

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

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the 41th Annual Conference of the Association for Computational Linguistics (ACL-03)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02)*.

Fei Xia, Martha Palmer, Nianwen Xue, Mary Ellen Okurowski, John Kovarik, Shizhe Huang, Tony Kroch, and Mitch Marcus. 2000. Developing Guidelines and Ensuring Consistency for Chinese Text Annotation. In *Proc. of the 2nd International Conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.

Jia Xu, Jianfeng Gao, Kristina Toutanova, and Hermann Ney. 2008. Bayesian semi-supervised chinese word segmentation for statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1017–1024, Manchester, UK, August. Coling 2008 Organizing Committee.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics and Chinese Language Processing*, volume 8, pages 29–48.

Hao Zhang, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008a. Bayesian learning of non-compositional phrases with synchronous parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 97–105, Columbus, Ohio.

Ruiqiang Zhang, Keiji Yasuda, and Eiichiro Sumita. 2008b. Improved statistical machine translation by multiple Chinese word segmentation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 216–223.

Synchronous Tree Adjoining Machine Translation

Steve DeNeefe and **Kevin Knight**
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292 USA
{sdeneefe,knight}@isi.edu

Abstract

Tree Adjoining Grammars have well-known advantages, but are typically considered too difficult for practical systems. We demonstrate that, when done right, adjoining improves translation quality without becoming computationally intractable. Using adjoining to model optionality allows general translation patterns to be learned without the clutter of endless variations of optional material. The appropriate modifiers can later be spliced in as needed.

In this paper, we describe a novel method for learning a type of Synchronous Tree Adjoining Grammar and associated probabilities from aligned tree/string training data. We introduce a method of converting these grammars to a weakly equivalent tree transducer for decoding. Finally, we show that adjoining results in an end-to-end improvement of +0.8 BLEU over a baseline statistical syntax-based MT model on a large-scale Arabic/English MT task.

1 Introduction

Statistical MT has changed a lot in recent years. We have seen quick progress from manually crafted linguistic models to empirically learned statistical models, from word-based models to phrase-based models, and from string-based models to tree-based models. Recently there is a swing back to incorporating more linguistic information again, but this time linguistic insight carefully guides the setup of empirically learned models.

Shieber (2007) recently argued that probabilistic Synchronous Tree Adjoining Grammars (Shieber and Schabes, 1990) have the right combination of properties that satisfy both linguists and empirical MT practitioners. So far, though, most work in this area has been either more linguistic than statistical (Abeille et al., 1990) or statistically-based, but linguistically light (Nesson et al., 2006).

Current tree-based models that integrate linguistics and statistics, such as GHKM (Galley et al., 2004), are not able to generalize well from a single phrase pair. For example, from the data in Figure 1, GHKM can learn rule (a) to translate nouns with two pre-modifiers, but does not generalize to learn translation rules (b) - (d) without the optional adjective or noun modifiers. Likewise, none of these rules allow extra material to be introduced, e.g. "Pakistan's national defense minister". In large enough training data sets, we see many examples of all the common patterns, but the rarer patterns have sparse statistics or poor coverage.

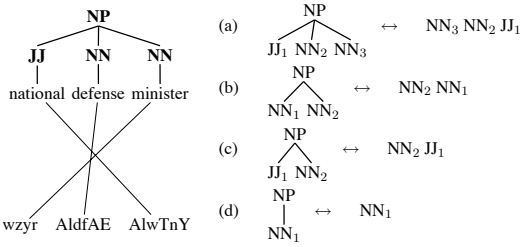


Figure 1: Rule (a) can be learned from this training example. Arguably, the more general rules (b) - (d) should also be learnable.

To mitigate this problem, the parse trees used as training data for these systems can be binarized (Wang et al., 2007). Binarization allows rules with partial constituents to be learned, resulting in more general rules, richer statistics, and better phrasal coverage (DeNeefe et al., 2007), but no principled required vs. optional decision has been made. This method's key weakness is that binarization always keeps adjacent siblings together, so there is no way to group the head with a required complement if optional information intervenes between the two. Furthermore, if all kinds of children are considered equally optional, then we have removed important syntactic constraints, which may end up permitting too much freedom. In addition, spurious alignments may limit the binarization tech-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 727–736,
Singapore, 6-7 August 2009.

nique's effectiveness.

In this paper, we present a method of learning a type of probabilistic Synchronous Tree Adjoining Grammar (STAG) automatically from a corpus of word-aligned tree/string pairs. To learn this grammar we use linguistic resources to make the required vs. optional decision. We then directly model the optionality in the translation rules by learning statistics for the required parts of the rule independently from the optional parts. We also present a method of converting these rules into a well-studied tree transducer formalism for decoding purposes. We then show that modeling optionality using adjoining results in a statistically significant BLEU gain over our baseline syntax-based model with no adjoining.

2 Translation Model

2.1 Synchronous Tree Insertion Grammars

Tree Adjoining Grammars (TAG), introduced by Joshi et al. (1975) and Joshi (1985), allow insertion of unbounded amounts of material into the structure of an existing tree using an adjunction operation. Usually they also include a substitution operation, which has a 'fill in the blank' semantics, replacing a substitution leaf node with a tree. Figure 2 visually demonstrates TAG operations. Shieber and Schabes (1990) offer a synchronous version of TAG (STAG), allowing the construction of a pair of trees in lockstep fashion using the TAG operations of substitution and adjunction on tree *pairs*. To facilitate this synchronous behavior, links between pairs of nodes in each tree pair define the possible *sites* for substitution and adjunction to happen. One application of STAG is machine translation (Abeille et al., 1990).

One negative aspect of TAG is the computational complexity: $O(n^6)$ time is required for monolingual parsing (and thus decoding), and STAG requires $O(n^{12})$ for bilingual parsing (which might be used for training the model directly on bilingual data). Tree Insertion Grammars (TIG) are a restricted form of TAG that was introduced (Schabes and Waters, 1995) to keep the same benefits as TAG (adjoining of unbounded material) without the computational complexity—TIG parsing is $O(n^3)$. This reduction is due to a limitation on adjoining: auxiliary trees can only introduce tree material to the left or the right of the node adjoined to. Thus an auxiliary tree can be classified by direction as left or right adjoining.

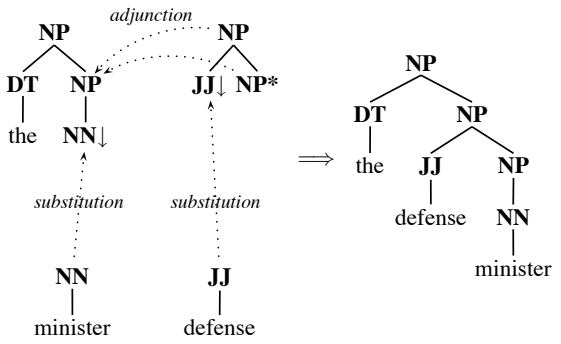


Figure 2: TAG grammars use substitution and adjunction operations to construct trees. Substitution replaces the substitution node (marked with ↓) with another tree. Adjunction inserts an auxiliary tree—a special kind of tree fragment with a foot node (marked with *)—into an existing tree at a permitted non-terminal node. Note that in TAG, adjunctions are permitted at *any* non-terminal with the same label as the root and foot node of the auxiliary tree, while in STAG adjunctions are restricted to linked sites.

Nesson et al. (2006) introduce a probabilistic, synchronous variant of TIG and demonstrate its use for machine translation, showing results that beat both word-based and phrase-based MT models on a limited-vocabulary, small-scale training and test set. Training the model uses an $O(n^6)$ bilingual parsing algorithm, and decoding is $O(n^3)$. Though this model uses trees in the formal sense, it does not create Penn Treebank (Marcus et al., 1993) style linguistic trees, but uses only one non-terminal label (**X**) to create those trees using six simple rule structures.

The grammars we use in this paper share some properties in common with those of Nesson et al. (2006) in that they are of the probabilistic, synchronous tree-insertion variety. All pairs of sites (both adjunction and substitution in our case) are explicitly linked. Adjunction sites are restricted by direction: at each linked site, the source and target side each specify one allowed direction. The result is that each synchronous adjunction site can be classified into one of four direction classes: {LR, LL, RR, RL}. For example, LR means the source side site only allows left adjoining trees and the target side site only allows right adjoining trees.

There are several important differences between our grammars and the ones of Nesson et al. (2006):

Richer, Linguistic Trees: Our grammars have a

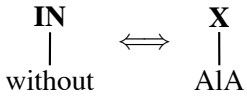
Penn Treebank-style linguistic tree on the English (target) side, and a hierarchical structure using only a single non-terminal symbol (**X**) on the source side. We believe this provides the rich information needed in the target language without over-constraining the model.

Substitution Sites/Non-lexical trees: We use both substitution and adjunction (Nesson et al. (2006) only used adjunction) and do not require all trees to contain lexical items as is commonly done in TIG (Schabes and Waters, 1995).

Single Adjunction/Multiple Sites: Each non-terminal node in a tree may allow multiple adjunction sites, but every site only allows at most one adjunction,[1] a common assumption for TAG as specified in the Vijay-Shanker (1987) definition.

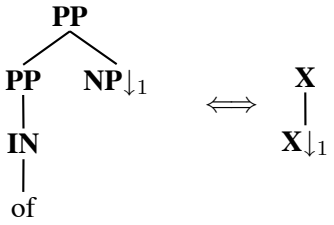
Here are some examples of automatically learned translation rules with interpretations of how they work:

1. simple lexical rules for translating words or phrases:



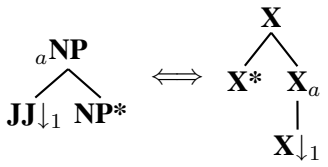
interpretation: translate the Arabic word "AlA" as the preposition "without"

2. rules with substitution for translating phrases with holes (substitution sites are designated by an arrow and numeric subscript, e.g. **NP**$\downarrow_1$):



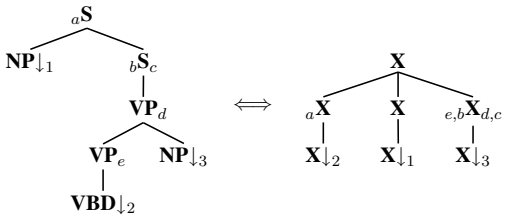
interpretation: insert "of" to turn a noun phrase into a prepositional phrase

3. simple adjoining rules for inserting optional modifiers (adjoining sites are designated by an alphabetic subscript before *or* after a non-terminal to indicate direction of adjoining, e.g. $_a$**NP**):



interpretation: adjoin an adjective before a noun in English but after in Arabic, and allowing further adjoinings in those same directions afterward

4. rules with multiple adjunction and substitution sites:



interpretation: translate an Arabic sentence in VSO form into an English sentence in SVO form, with multiple adjoining options

2.2 Generative Story

When we use these rules to translate from a foreign sentence f into an English sentence e, we use several models together in a log-linear fashion, but our primary model is a joint model of $P(e_{tree}, f_{tree})$, which is our surrogate for directly modeling $P(e|f)$. This can be justified because $P(e|f) = \frac{P(e,f)}{P(f)}$, and $P(f)$ is fixed for a given foreign sentence. Therefore:

$$\begin{aligned}\underset{e}{\operatorname{argmax}} P(e|f) &= \underset{e}{\operatorname{argmax}} P(e,f) \\ &\approx \operatorname{yield}(\underset{e_{tree}}{\operatorname{argmax}} P(e_{tree}, f_{tree})) \\ &\approx \operatorname{yield}(\underset{e_{tree}}{\operatorname{argmax}} P(d_{e_{tree},f_{tree}}))\end{aligned}$$

where $d_{e_{tree},f_{tree}}$ is a derivation tree of rules that generates e_{tree} and f_{tree}. In other words, e, the highest probability translation of f, can be approximated by taking the yield of the highest probability tree e_{tree} that is a translation of the highest probability tree of f. This can further be approximated by the highest probability derivation of rules translating between f and e via trees.

Now we define the probability of generating $d_{e_{tree},f_{tree}}$. Starting with an initial symbol pair

[1] An adjoined rule may itself have adjoining sites allowing further adjunction.

representing a rule with a single substitution site,[2] $\langle$TOP$\downarrow$, X$\downarrow\rangle$, a tree pair can be generated by the following steps:

1. For each substitution site s_i in the current rule r_1:
   (a) Choose with probability $P_{sub}(r_2|\langle \text{label}_L(s_i), \text{label}_R(s_i)\rangle)$ a rule r_2 having root node labels $\text{label}_L(s_i)$ and $\text{label}_R(s_i)$ that match the left and right labels at s_i.
2. For each adjunction site s_{i,r_1} in the current rule r_1:
   (a) Choose with rule-specific probability $P_{\text{if}adj}(\text{decision}_{adjoin}|s_{i,r_1}, r_1)$ choose whether or not to adjoin at the current site s_{i,r_1}.
   (b) If we *are* adjoining at site s_{i,r_1}, choose with probability $P_{adj}(r_2|d, \langle \text{label}_L(s_{i,r_1}), \text{label}_R(s_{i,r_1})\rangle)$ a rule r_2 of direction class d having root node labels $\text{label}_L(s_{i,r_1})$ and $\text{label}_R(s_{i,r_1})$ that match the left and right labels at s_{i,r_1}.
3. Recursively process each of the added rules

For all substitution rules r_s, adjoining rules r_a, and adjoining sites $s_{i,r}$, the probability of a derivation tree using these rules is the product of all the probabilities used in this process, i.e.:

$$P_{deriv} = \prod_{r_s} \left(P_{sub}(r_s|\langle \text{root}_L(r_s), \text{root}_R(r_s)\rangle) \cdot \prod_{s_{i,r_s}} P_{\text{if}adj}(\text{decision}_{adjoin}|s_{i,r_s}, r_s) \right) \cdot \prod_{r_a} \left(P_{adj}(r_a|\text{dir}(r_a), \langle \text{root}_L(r_a), \text{root}_R(r_a)\rangle) \cdot \prod_{s_{i,r_a}} P_{\text{if}adj}(\text{decision}_{adjoin}|s_{i,r_a}, r_a) \right)$$

Note that while every new substitution site requires an additional rule to be added, adjunction sites may or may not introduce an additional rule based on the rule-specific $P_{\text{if}adj}$ probability. This allows adjunction to represent linguistic optionality.

[2]Here and in the following, we use *site* as shorthand for synchronous site pair.

3 Learning the Model

Instead of using bilingual parsing to directly train our model from strings as done by Nesson et al. (2006), we follow the method of Galley et al. (2004) by dividing the training process into steps. First, we word align the parallel sentences and parse the English (target) side. Then, we transform the aligned tree/string training data into derivation trees of minimal translation rules (Section 3.1). Finally, we learn our probability models P_{sub}, $P_{\text{if}adj}$, and P_{adj} by collecting counts over the derivation trees (Section 3.2). This method is quick enough to allow us to scale our learning process to large-scale data sets.

3.1 Generating Derivation Trees and Rules

There are four steps in transforming the training data into derivation trees and rules, the first two operating only on the English parse tree itself:[3]

A. Marking Required vs. Optional. For each constituent in the English parse tree, we mark children as (H)ead, (R)equired, or (O)ptional elements (see step (a) in Figure 3). The choice of head, required, or optional has a large impact on the generality and applicability of our grammar. If all children are considered required, the result is the same as the GHKM rules of Galley et al. (2004) and has the same problem—lots of low count, syntactically over-constrained rules. Too many optional children, on the other hand, allows ungrammatical output. Our proposed model is a linguistically motivated middle ground: we consider the linguistic heads and complements selected by Collins' (2003) rules to be required and all other children to be optional.

B. Parse tree to TIG tree. Next, we restructure the English tree to form a TIG derivation where head and required elements are substitutions, and optional elements are adjunctions (see step (b) in Figure 3). To allow for adjoining between siblings under a constituent, we first do a head-out binarization of the tree. This is followed by excising[4] any children marked as optional and replacing them with an adjunction site, as shown in Figure 4. Note that we excise a chain of optional children as one site with each optional child

[3]These first two steps were inspired by the method Chiang (2003) used to automatically extract a TIG from an English parse tree.

[4]Excising is the opposite of adjoining: extracting out an auxiliary rule from a tree to form two smaller trees.

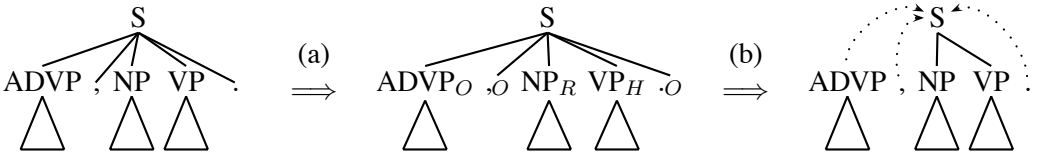
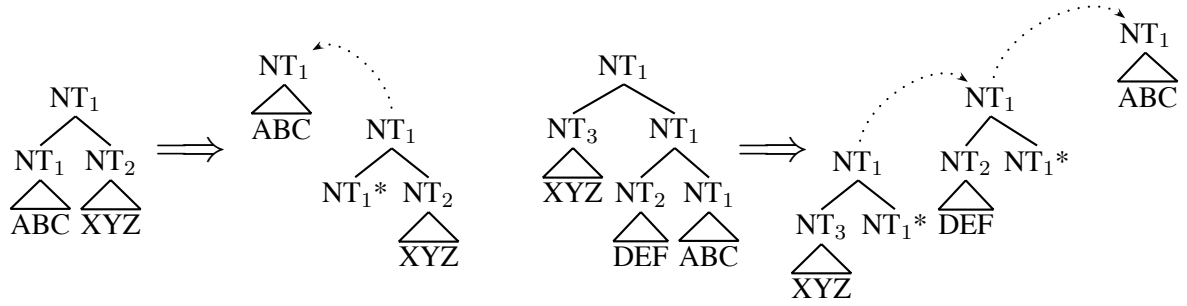


Figure 3: Parse tree to TIG transformation: (a) mark constituent children with (H)ead, (R)equired, and (O)ptional, then (b) restructure the tree so that head and required elements are substitutions, while optional elements are adjoined (shown with dotted lines).



(a) excising one optional child (XYZ) (b) excising a series of optional children (DEF, then XYZ)

Figure 4: Two examples of excising auxiliary trees from a head-out binarized parse tree: (a) excising one optional left branch, (b) excising a chain of optional branches in the same (right) direction into a series of adjunctions. In both examples, the 'ABC' child is the head, while the other children are optional.

adjoined to the previous child, as in Figure 4(b).

C. Extracting rules and derivation trees. We now have a TIG derivation tree, with each elementary tree attached to its parent by a substitution or adjunction link. We can now extract synchronous rules allowed by the alignments and syntactic constituents. This can be done using a method inspired by the rule-extraction approach of Galley et al. (2004), but instead of directly operating on the parse tree we process the English TIG derivation tree. In bottom-up fashion, we visit each elementary tree in the derivation, allowing a rule rooted at this tree to be extracted if its words or those of its descendants are aligned such that they are the English side of a self-contained parallel phrase (i.e., the foreign text of this phrase is not aligned to English leaves outside of the set of descendants). Otherwise, this elementary tree is rejoined with its parent to form a larger elementary tree. At the end of this process we have a new set of linked elementary trees which make up the English side of the grammar, where each substitution or adjunction link becomes a substitution or adjunction site in the synchronous grammar.

On the foreign side we start with the foreign text of the self-contained parallel phrase and replace any parts of this phrase covered by substituted or adjoined children of the English side tree with substitution sites or adjunction site markers. From this, we produce a tree with a simple, regular form by placing all items under a root node labeled X. In the case of more than one foreign word or substitution site, we introduce an intermediate level of X-labeled non-terminals to allow for possible adjunction between elements, otherwise the adjoining sites attach to the single root node. We attach all foreign-side adjoining sites to be left adjoining, except on the right side of the right-hand child.

It is possible to have the head child tree on the English side not aligned to anything, while the adjoined children are. This may lead to rules with no foreign non-terminal from which to anchor the adjunctions, so in this case, we attach adjoined child elementary trees starting from the head and moving out until we attach a some child with a non-empty foreign side.

D. Generalizing rules. We need to clarify what makes one rule distinct from another. Consider the example in Figure 5, which shows selected rules learned in the case of two different noun phrases. If the noun phrase consists of just a single noun, we learn rule (a), while if the noun phrase also has an adjective, we learn rules (b) and (c). Since adjoining the adjective is optional, *we*

consider rules (a) and (c) to be the same rule, the latter with an adjoining seen, and the former with the same adjoining not seen.

3.2 Statistical Models

Once we have the derivation trees and list of rules, we learn our statistical models using maximum likelihood estimation. By counting and normalizing appropriately over the entire corpus, we can straightforwardly learn the P_{sub} and P_{adj} distributions. However, recall that in our model $P_{\text{if}adj}$ is a rule-specific probability, which makes it more difficult to estimate accurately. For common rules, we see plenty of examples of adjoining, while for other rules, we need to learn from only a handful of examples. Smoothing and generalization are especially important for these low frequency cases.

Two options present themselves for how to estimate adjoining:

(a) A joint model of adjoining. We assume that adjoining decisions are made in combination with each other, and so learn non-zero probabilities only for adjoining combinations seen in data

(b) An independent model of adjoining. We assume adjoining decisions are made independently, and learn a model for each adjoining site separately

Option (a) may be sufficient for frequent rules, and will accurately model dependencies between different kinds of adjoining. However, it does not allow us to generalize to unseen patterns of adjoining. Consider the low frequency situation depicted in Figure 6, rules (d)-(f). We may have seen this rule four times, once with adjoining site a, twice with adjoining sites a and b, and once with a third adjoining site c. The joint model will give a zero probability to unseen patterns of adjoining, e.g. no adjoining at any site or adjoining at site b alone. Even if we use a discounting method to give a non-zero probability to unseen cases, we still have no way to distinguish one from another.

Option (b) allows us to learn reasonable estimates for these missing cases by separating out adjoining decisions and letting each speak for itself. To properly learn non-zero probabilities for unseen cases[5] we use add k smoothing ($k = \frac{1}{2}$).

[5]For example, low frequency rules may have always been observed with a single adjoining pattern, and never without adjoining.

A weakness of this approach still remains: adjoining is not a truly independent process, as we observe empirically in the data. In real data, frequent rules have many different observed adjoining sites (10 or 20 in some cases), many of which represent already infrequent sites in combinations never seen together. To reduce the number of invalid combinations produced, we only allow adjoinings to be used at the same time if they have occurred together in the training data. This restriction makes it possible to do less adjoining than observed, but not more. For the example in Figure 6, in addition to the observed patterns, we would also allow site b to be used alone, and we would allow no adjoinings, but we would not allow combinations of site c with either a or b. Later, we will see that this makes the decoding process more efficient.

Because both option (a) and (b) above have strengths and weaknesses, we also explore a third option which builds upon the strengths of each:

(c) A log-linear combination of the joint model and independent model. We assume the probability has both a dependent and independent element, and learn the relative weight between them automatically

To help smooth this model we add two additional binary features: one indicating adjoining patterns seen in data and one indicating previously unseen patterns.

4 Decoding

To translate with these rules, we do a monolingual parse using the foreign side of the rules (constraining the search using non-terminal labels from both sides), while keeping track of the English side string and structure for language modeling purposes. This produces all valid derivations of rules whose foreign side yield is the input string, from which we simply choose the one with the highest log-linear model score. Though this process could be done directly using a specialized parsing algorithm, we note that these rules have weakly equivalent counterparts in the Synchronous Tree Substitution Grammar (STSG) and Tree-to-string transducer (xLNTs[6]) worlds, such that each STIG rule can be translated into one equivalent rule, plus some helper rules to model the adjoin/no-adjoin

[6]xLNTs is shorthand for extended linear non-deleting top-down tree-to-string transducer.

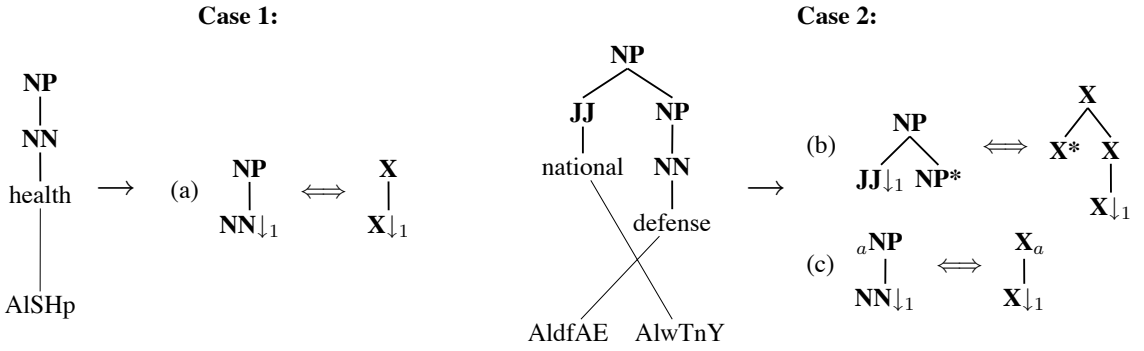
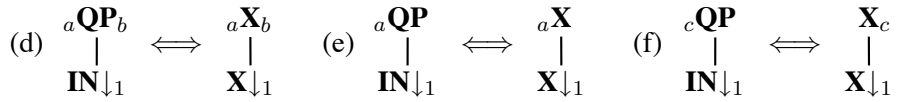


Figure 5: Selected rules learned in two cases. Rule (a) and (c) are considered the same rule, where (c) has the optional synchronous adjoining site marked with a. From these (limited) examples alone we would infer that adjective adjoining happens half the time, and is positioned before the noun in English, but after the noun in Arabic (thus the positioning of site a).



(seen once) (seen twice) (seen once)

Figure 6: For a low frequency rule, we may see only a few different adjoining patterns, but we want to infer more.

decision. Conversion to a better known and explored formalism allows us to take advantage of existing code and algorithms. Here we describe the conversion process to xLNTs rules, though conversion to STSG is similar.

Algorithm 1 describes the process of converting one of our automatically learned STIG rules. On each side of the rule, we traverse the tree in a top-down, left-to-right order, recording words, substitution sites, and adjoining sites in the order encountered (left adjoinings before the node's children and right adjoinings after). We make these words and sites as the children under a single root node. The substitution sites are given states made up of a combination of their source and target labels as are the roots of non-adjoining rules. Adjoining sites are labeled with a combination of the rule id and a site id. Adjoining rule roots are labeled with a combination of the source and target root labels and the direction class. To allow for the adjoining/no-adjoining decision, two helper rules are created for each adjoining site, their root state a combination of the rule and site ids. One of these rules has only epsilon leaf nodes (representing no adjoining), while the other has leaf nodes and a state that match with the corresponding adjoining rule root (labeled with the site's source and target labels and the direction class).

For each rule, the algorithm generates one main rule and pairs of helper rules to facilitate adjoining/non-adjoining. For computational efficiency reasons, our decoder supports neither epsilon rules nor non-binary rules. So we remove epsilons using an exponential expansion of the rules: combine each main rule with an adjoining or non-adjoining helper rule for each adjunction site, then remove epsilon-only branches. For k adjunction sites this could possibly results in 2^k rules. But as discussed previously (at the end of Section 3.2), we only allow subsets of adjoining combinations seen in training data, so this number is substantially lower for large values of k.

5 Experiments

All experiments are trained with a subset (171,000 sentences or 4 million words) of the Arabic-English training data from the constrained data track of the NIST 2008 MT Evaluation, leaving out LDC2004T18, LDC2007E07, and the UN data. The training data is aligned using the LEAF technique (Fraser and Marcu, 2007). The English side of the training data is parsed with an implementation of Collins Model 2 (Collins, 2003) then head-out binarized. The tuning data (1,178 sentences) and devtest data (1,298 sentences) are

Input: Synchronous TIG rule r with j adjoining sites, $S \leftrightarrow T$, where S and T are trees
Output: a weakly equivalent xLNTs rule $S' \leftrightarrow t_1 \ldots t_n$, where S' is a one-level tree, and $2 \cdot j$ helper rules for adjoining
Run time: $O(|S| + |T|)$
begin
  rules $\leftarrow \{\}$, lhs-state $\leftarrow$ concat('q', get-root(S), get-root(T))
  site-and-word-list-s $\leftarrow$ get-sites-and-words-in-order(S)
  site-and-word-list-t $\leftarrow$ get-sites-and-words-in-order(T)
  if *r is adjoining* **then** lhs-state $\leftarrow$ concat(lhs-state, get-adjoin-dir(S), get-adjoin-dir(T))
  lhs $\leftarrow$ construct-LHS(lhs-state, get-root(S), site-and-word-list-s)
  rhs $\leftarrow$ construct-RHS(add-states(id(r), site-and-word-list-t))
  add(rules, 'lhs $\leftrightarrow$ rhs') `/* main rule */`
  foreach *adjoining site* $i \in 1 \ldots k$ **do**
    lhs-state $\leftarrow$ concat('q', id(r), i), rhs-state $\leftarrow$ concat('q', lhs-root)
    lhs-root $\leftarrow$ concat(source-label(i), target-label(i), source-dir(i), target-dir(i))
    lhs $\leftarrow$ construct-LHS(lhs-state, lhs-root, lhs-root)
    rhs $\leftarrow$ construct-RHS({(rhs-state, lhs-root)})
    rhs-eps $\leftarrow$ construct-RHS(ϵ)
    add(rules, {'lhs $\leftrightarrow$ rhs', 'lhs $\leftrightarrow$ rhs-eps'}) `/* helper rules for site` i `*/`
  return rules
end
function *get-sites-and-words-in-order(node)*
  $y \leftarrow \{\}$
  if *node is substitution site or word* **then** append site or word to y **else**
    append left adjoining sites to y in outside-to-inside order
    foreach *child c of node* **do** append result of get-yield(c) to y
    append right adjoining sites to y in inside-to-outside order
  return y
end
function *add-states(ruld-id, node-list)*
  foreach *substitution or adjunction site s_i and in node-list* **do**
    if *s_i is substitution site* **then** state = concat('q', source-site-label(s_i), target-site-label(s_i))
    else state = concat('q', rule-id, i)
    replace s_i with (state, s_i)
  return modified node-list
end

Algorithm 1: Conversion from synchronous TIG rules to weakly equivalent xLNTs rules

		BLEU	
	description	**DevTest**	**NIST06**
(1)	baseline: all required (GHKM minimal, head-out binarized parse trees)	48.0	47.0
(2)	joint adjoining prob model alone (only observed adjoining patterns)	48.0	46.6
(3)	independent adjoining prob model alone (only observed adjoining patterns)	48.1	46.7
(4)	independent adjoining prob model alone (with new adjoining patterns)	48.5	47.6
(5)	independent model alone + features (adjoining pattern, direction)	48.4	47.7
(6)	log-linear combination of joint & independent models + features	48.7	47.8

Table 1: End-to-end MT results show that the best adjoining model using a log-linear combination of joint and independent models (line 6) outperforms the baseline (line 1) by +0.7 and +0.8 BLEU, a statistically significant difference at the 95% confidence level.

made up of newswire documents drawn from the NIST MT evaluation data from 2004, 2005, and 2006 (GALE part). We use the newswire documents from the NIST part of the 2006 evaluation data (765 sentences) as a held-out test set.

We train our feature weights using max-BLEU (Och, 2003) and decode with a CKY-based decoder that supports language model scoring directly integrated into the search.

In addition to P_{sub}, P_{adj}, and $P_{\text{if}adj}$, we use several other features in our log-linear model during decoding, including: lexical and phrase-based translation probabilities, a model similar to conditional probability on the trees ($P(f_{tree}(rule)|e_{tree}(rule))$), a probability model for generating the top tree non-terminal, a 5-gram language model[7], and target length bonus. We also have several binary features—lexical rule, rule with missing or spurious content words—and several binary indicator features for specialized rules: unknown word rules; name, number, and date translation rules; and special fail-safe monotone translation rules in case of parse failures and extremely long sentences.

Table 1 shows the comparison between our baseline model (minimal GHKM on head-out binarized parse trees) and different models of adjoining, measured with case-insensitive, NIST-tokenized BLEU (IBM definition). The top section (lines 1–4) compares the joint adjoining probability model to the independent adjoining probability model and seen vs. unseen adjoining combinations. While the joint model results in a BLEU score at the same level as our baseline (line 2), the independent model (line 4) improves BLEU by +0.5 and +0.6, which are significant differences at the 95% confidence level. Since with the independent model we introduce both new adjoining patterns and a different probability model for adjoining (each site is independent), we also use the independent model with only previously seen adjoining patterns (line 3). The insignificant difference in BLEU between lines 2 and 3 leads us to think that the new adjoining patterns are where the improvement comes from, rather than the independent probability model alone.

We also test several other features and combinations. First, we add binary features to indicate a new adjoining combination vs. one previously seen in data. We also add features to indicate the direction class of adjoining to test if there is a systematic bias toward particular directions. These features cause no significant difference in score (line 5). We also add the joint-adjoining probability as a feature, allowing it to be combined in a log-linear fashion with the independent probability (line 6). This results in our best BLEU gain: +0.7 and +0.8 over our non-adjoining baseline.

[7]The 5-gram LM was trained on 2 billion words of automatically selected collections taken from the NIST 08 allowable data.

6 Conclusion

We have presented a novel method for learning the rules and probabilities for a new statistical, linguistically-informed, syntax-based MT model that allows for adjoining. We have described a method to translate using this model. And we have demonstrated that linguistically-motivated adjoining improves the end-to-end MT results.

There are many potential directions for research to proceed. One possibility is to investigate other methods of making the required vs. optional decision, either using linguistic resources such as COMLEX or automatically learning the distinction using EM (as done for tree binarization by Wang et al. (2007)). In addition, most ideas presented here are extendable to rules with linguistic trees on both sides (using insights from Lavie et al. (2008)). Also worth investigating is the direct integration of bilingual dictionaries into the grammar (as suggested by Shieber (2007)). Lastly, rule composition and different amounts of lexicalization (Galley et al., 2006; Marcu et al., 2006; DeNeefe et al., 2007) or context modeling (Mariño et al., 2006) have been successful with other models.

Acknowledgments

We thank David Chiang for suggestions about adjoining models, Michael Pust and Jens-Sönke Vöckler for developing parts of the experimental framework, and other colleagues at ISI for their helpful input. We also thank the anonymous reviewers for insightful comments and suggestions. This research is financially supported under DARPA Contract No. HR0011-06-C-0022, BBN subcontract 9500008412.

References

Anne Abeille, Yves Schabes, and Aravind K. Joshi. 1990. Using lexicalized TAGs for machine translation. In *Proc. COLING*, volume 3.

David Chiang. 2003. Statistical parsing with an automatically extracted tree adjoining grammar. *Data-Oriented Parsing*.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4).

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proc. EMNLP-CoNLL*.

Alexander Fraser and Daniel Marcu. 2007. Getting the structure right for word alignment: LEAF. In *Proc. EMNLP-CoNLL*.

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

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

Aravind K. Joshi, L. S. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1).

Aravind K. Joshi. 1985. How much context-sensitivity is necessary for characterizing structural descriptions—tree adjoining grammars. *Natural Language Processing—Theoretical, Computational, and Psychological Perspectives*.

Alon Lavie, Alok Parlikar, and Vamshi Ambati. 2008. Syntax-driven learning of sub-sentential translation equivalents and translation rules from parsed parallel corpora. In *Proc. SSST*.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. EMNLP*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2).

José B. Mariño, Rafael E. Banchs, Josep M. Crego, Adrià de Gispert, Patrik Lambert, José A. R. Fonollosa, and Marta R. Costa-jussà. 2006. N-gram-based machine translation. *Computational Linguistics*, 32(4).

Rebecca Nesson, Stuart M. Shieber, and Alexander Rush. 2006. Induction of probabilistic synchronous tree-insertion grammars for machine translation. In *Proc. AMTA*.

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

Yves Schabes and Richard C. Waters. 1995. Tree insertion grammar: A cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21(4).

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proc. COLING*.

Stuart M. Shieber. 2007. Probabilistic synchronous tree-adjoining grammars for machine translation: The argument from bilingual dictionaries. In *Proc. SSST Wkshp., NAACL-HLT*.

Kumar Vijay-Shanker. 1987. *A study of tree adjoining grammars*. Ph.D. thesis.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proc. EMNLP and CoNLL*.

Word Buffering Models for Improved Speech Repair Parsing*

Tim Miller
University of Minnesota – Twin Cities
tmill@cs.umn.edu

Abstract

This paper describes a time-series model for parsing transcribed speech containing disfluencies. This model differs from previous parsers in its explicit modeling of a buffer of recent words, which allows it to recognize repairs more easily due to the frequent overlap in words between errors and their repairs. The parser implementing this model is evaluated on the standard Switchboard transcribed speech parsing task for overall parsing accuracy and edited word detection.

1 Introduction

Speech repair is a phenomenon in spontaneous speech where a speaker interrupts the flow of speech (at what's called the *interruption point*), backtracks some number of words (the *reparandum*), and continues the utterance with material meant to replace the reparandum (the *alteration*).[1] The utterance can be rendered syntactically correct by excising all the words that the speaker skipped over when backtracking. Speech with repair is difficult for machines to process because in addition to detecting repair, a system must know what words are meant to be excised, and parsing systems must determine how to form a grammatical structure out of the set of words comprising both the error speech and the correct speech.

Recent approaches to syntactic modeling of speech with repairs have shown that significant gains in parsing accuracy can be achieved by modeling the syntax of repairs (Hale et al., 2006; Core and Schubert, 1999). In addition, others have shown that a parser based on a time-series model that explicitly represents the incomplete constituents in fluent and disfluent speech can also improve parsing accuracy (Miller and Schuler, 2008). However, these parsing approaches are still not as accurate at detecting reparanda as classification systems which use a variety of features to detect repairs (Charniak and Johnson, 2001; Johnson and Charniak, 2004; Heeman and Allen, 1999).

One highly salient feature which classification systems use to detect repair is the repetition of words between the error and the repair. Johnson and Charniak report that 60% of words in the alterations are copies of words in reparanda in the Switchboard corpus. Typically, this information is not available to a parser trained on context-free grammars.

Meanwhile, psycholinguistic models suggest that the human language system makes use of buffers both to keep track of recent input (Baddeley et al., 1998) and to smooth out generation (Levelt, 1989). These buffers are hypothesized to contain representations of recent phonological events, suggesting that there is a short window where new input might be compared to recent input. This could be represented as a buffer which predicts or detects repeated input in certain constrained circumstances.

This paper describes a hybrid parsing system operating on transcribed speech which combines an incremental parser implemented as a probabilistic time-series model, as in Miller and Schuler, with a buffer of recent words meant to loosely model something like a phonological loop, which should better account for word repetition effects in speech repair.

2 Background

This work uses the Switchboard corpus (Godfrey et al., 1992) for both training and testing. This corpus contains transcribed and syntactically annotated conversations between human interlocutors. The reparanda in speech repairs are ulti-

This research was supported by NSF CAREER award 0447685. The views expressed are not necessarily endorsed by the sponsors .

[1] This terminology follows Shriberg (1994).

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 737–745,
Singapore, 6-7 August 2009.

mately dominated by the EDITED label, and in cases where the reparandum ends with an unfinished constituent, the lowest constituent label is augmented with the -UNF tag. These annotations provide necessary but not sufficient information for parsing speech with repairs, and thus many improvements in performing this task come as the result of modifying these annotations in the training data.

As mentioned above, both Hale and colleagues (2006) and Miller and Schuler (2008) showed that speech repairs contain syntactic regularities, which can improve the parsing of transcribed speech with repairs when modeled properly. Hale et al. used 'daughter annotation', which adds the label of an EDITED node's child to the EDITED label itself, and '-UNF propagation', which labels every node between an original -UNF node and the EDITED with an -UNF tag. Miller and Schuler used a 'right-corner transform' to convert standard phrase structure trees of the Penn Treebank into 'right-corner trees', which have highly left-branching structure and non-standard tree categories representing incomplete constituents being recognized. These trees can be mapped into a fixed-depth Hierarchical Hidden Markov Model to achieve improved parsing and reparandum-finding results over standard CYK parsers.

Work by Johnson and Charniak (2004; 2001) uses much of the same structure, but is not a parsing approach per se. In earlier work, they used a boosting algorithm using word identity and category features to classify individual words as part of a reparandum or not, and achieved very impressive accuracy. More recent work uses a tree-adjoining grammar (TAG) to model the overlap in words and part-of-speech tags between reparandum and alteration as context sensitive syntax trees. A parser is then used to rank the multiple outputs of the TAG model with reparandum words removed.

Another approach that makes use of the correspondence between words in the reparandum and alteration is Heeman and Allen (1999). This approach uses several sources of evidence, including word and POS correspondence, to predict repair beginnings and correct them (by predicting how far back they are intended to retrace). This model includes random variables between words that correspond to repair state, and in a repair state, allows words in the reparandum to 'license' words in the alteration with high probability, accounting for the high percentage of copied words and POS tags between reparandum and alteration.

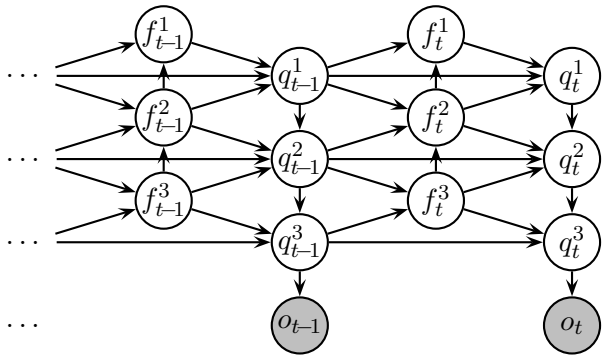


Figure 1: Graphical representation of the dependency structure in a standard Hierarchic Hidden Markov Model with $D = 3$ hidden levels that can be used to parse syntax. Circles denote random variables, and edges denote conditional dependencies. Shaded circles denote variables with observed values.

3 Model Description

This work is based on a standard Hierarchical Hidden Markov Model parser (Schuler, 2009), with the addition of two new random variables for tracking the state of speech repair. The HHMM framework is a desirable starting point for this work for two reasons: First, its definition in terms of a graphical model makes it easy to think about and to add new random variables. Second, the HHMM parser operates incrementally in a left-to-right fashion on word input, which allows this system to run in a single pass, conditioning current words on a hypothesized buffer and interruption point variable. The incremental nature of this system is a constraint that other systems are not bound by, but makes this model more psycholinguistically plausible. In comparison, a CYK parsing framework attempting to use the same probabilistic model of word dependency between reparanda and alterations would need to do a second pass after obtaining the most likely parses, in order to tell if a particular word's generation probability in a specific parse is influenced by a recent repair.

The graphical model representation of this framework is illustrated in Figures 1 and 4. The original model, shown in Figure 1, has complex variables Q and F broken down into several q^d_t and f^d_t for time step t and depth d. These ran-

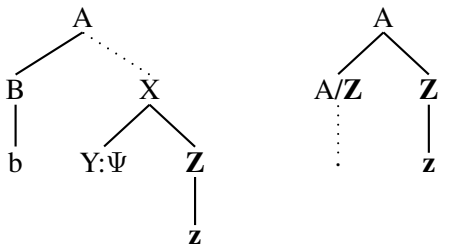
dom variables will be explained shortly, but for now suffice it to say that in this work they are unaltered from the original HHMM parsing framework, while those labeled I and B (Figure 4) are additions specific to the system described in this paper. This section will next describe the standard HHMM parsing framework, before describing how this work augments it.

3.1 Right-corner Transform

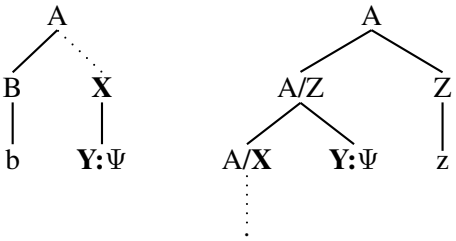
The HHMM parser consists of stacks of a fixed depth, which contain hypotheses of constituents that are being processed. In order to minimize the number of stack levels needed in processing, the phrase structure trees in the training set are modified using a 'right-corner transform', which converts right expansion in trees to left expansion, leaving heavily left-branching structure requiring little depth. The right-corner transform used in this paper is simply the left-right dual of a left-corner transform (Johnson, 1998a).

The right-corner transform can be defined as a recursive algorithm on phrase-structure trees in Chomsky Normal Form (CNF). Trees are converted to CNF first by binarizing using standard linguistically-motivated techniques (Klein and Manning, 2003; Johnson, 1998b). Remaining unbinarized structure is binarized in a brute force fashion, creating right-branching structure by creating a single node which dominates the two right-most children of a 'super-binary' tree, with the label being the concatenation of its children's labels (see Figure 2).

Taking this CNF phrase structure tree as input, the right-corner transform algorithm keeps track of two separate trees, the original and the new right-corner tree it is building. This process begins at the right-most preterminal of the original tree, and works its way up along the right 'spine', while building its way down a corresponding left spine of the new right-corner tree. The trees below shows the first step of the algorithm, with the tree on the left being disassembled, the tree on the right being built from its parts, and the working positions in the trees shown in bold.



The bottom right corner of the original tree is made the top right corner of the new tree, and the left corner of the new tree is made the new working position and given a 'slash' category A/Z. The 'slash' category label A/Z represents a tree that is the start of a constituent of type A that needs a right-child of type Z in order to complete. The new right-corner of the original tree is the parent (X) of the previous right corner, and its subtree is now added to the right-corner derivation:



After the first step, the subtrees moved over to the right-corner tree may have more complex substructure than a single word (in this case, Ψ represents that possibly complex structure). After being attached to the right-corner tree in the correct place, the algorithm is recursively applied to that now right-branching substructure.

Again, the left child is given a new slash category: The 'active constituent' (the left side of a slash category) is inherited from the root, and the 'awaited constituent' (the right side of a slash category) is taken from the constituent label of the right-corner it came from.

This algorithm proceeds iteratively up the right spine of the original tree, moving structure to the right-corner tree and recursively transforming it as it is added. The final step occurs when the original root (A in this case) is reduced to having a single child, in which case its child is added as a child of the leftmost current branch of the right-corner tree, and it is transformed recursively.

Figures 2 and 3 show an example tree from the Switchboard corpus before and after the right-corner transform is applied.

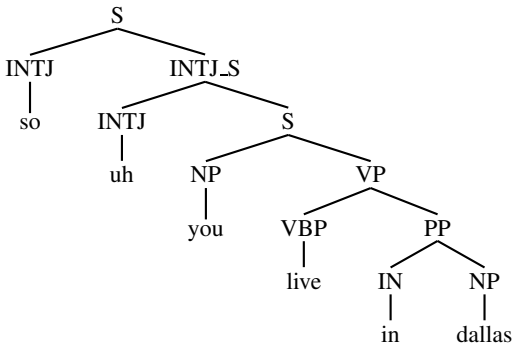


Figure 2: Input to the right-corner transform. This tree also shows an example of the 'brute-force' binarization done on super-binary branches that cannot be otherwise be binarized with linguistically-motivated rules.

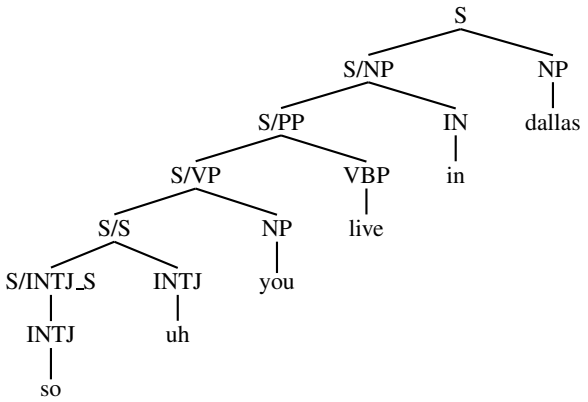


Figure 3: Right-corner transformed version of the tree in Figure 2.

3.2 Hierarchical Hidden Markov Model

A Hierarchical Hidden Markov Model is essentially an HMM with a specific factorization that is useful in many domains — the hidden state at each time step is factored into d random variables which function as a stack, and d additional random variables which regulate the operations of the stack through time. For the model of speech repair presented here, an interruption point is identified by one of these regulator variables firing earlier than it would in fluent speech. This concept will be formalized below. The stack regulating random variables are typically marginalized out when performing inference on a sequence.

While the vertical direction of the hidden sub-states (at a fixed t) represents a stack at a single point in time, the horizontal direction of the hidden sub-states (at a fixed d) can be viewed as a simple HMM at depth d, expanding the state from the HMM above it across multiple time steps and causing the HMM below it to expand its own states. This interpretation will be useful when formally defining the transitions between the stack elements at different time steps below.

Formally, HMMs characterize speech or text as a sequence of hidden states q_t (which may consist of speech sounds, words, and/or other hypothesized syntactic or semantic information), and observed states o_t at corresponding time steps t (typically short, overlapping frames of an audio signal, or words or characters in a text processing application). A most likely sequence of hidden states $\hat{q}_{1..T}$ can then be hypothesized given any sequence of observed states $o_{1..T}$, using Bayes' Law (Equation 2) and Markov independence assumptions (Equation 3) to define a full $\mathsf{P}(q_{1..T} \mid o_{1..T})$ probability as the product of a *Language Model* (Θ_{L}) prior probability and an *Observation Model* (Θ_{O}) likelihood probability:

$$\hat{q}_{1..T} = \underset{q_{1..T}}{\operatorname{argmax}}\, \mathsf{P}(q_{1..T} \mid o_{1..T}) \tag{1}$$

$$= \underset{q_{1..T}}{\operatorname{argmax}} \mathsf{P}(q_{1..T}) \cdot \mathsf{P}(o_{1..T} \mid q_{1..T}) \tag{2}$$

$$\stackrel{\text{def}}{=} \underset{q_{1..T}}{\operatorname{argmax}} \prod_{t=1}^{T} \mathsf{P}_{\Theta_{\mathrm{L}}}(q_t \mid q_{t-1}) \cdot \mathsf{P}_{\Theta_{\mathrm{O}}}(o_t \mid q_t) \tag{3}$$

Language model transitions $\mathsf{P}_{\Theta_{\mathrm{L}}}(q_t \mid q_{t-1})$ over complex hidden states q_t can be modeled using synchronized levels of stacked-up component HMMs in a Hierarchic Hidden Markov Model (HHMM) (Murphy and Paskin, 2001). HHMM transition probabilities are calculated in two phases: a 'reduce' phase (resulting in an intermediate, marginalized state f_t), in which component HMMs may terminate; and a 'shift' phase (resulting in a modeled state q_t), in which unterminated HMMs transition, and terminated HMMs are re-initialized from their parent HMMs. Variables over intermediate f_t and modeled q_t states are factored into sequences of depth-specific variables — one for each of D levels in the HMM hierarchy:

$$f_t = \langle f_t^1 \dots f_t^D \rangle \tag{4}$$

$$q_t = \langle q_t^1 \dots q_t^D \rangle \tag{5}$$

Transition probabilities are then calculated as a product of transition probabilities at each level, using level-specific 'reduce' Θ_{F} and 'shift' Θ_{Q} mod-

els:

$$\mathsf{P}_{\Theta_{\text{L}}}(q_t|q_{t\text{-}1}) = \sum_{f_t} \mathsf{P}(f_t|q_{t\text{-}1}) \cdot \mathsf{P}(q_t|f_t\, q_{t\text{-}1}) \quad (6)$$

$$\stackrel{\text{def}}{=} \sum_{f_t^{1..D}} \prod_{d=1}^{D} \mathsf{P}_{\Theta_{\text{F}}}(f_t^d|\, f_t^{d+1} q_{t\text{-}1}^d q_{t\text{-}1}^{d-1}) \cdot \mathsf{P}_{\Theta_{\text{Q}}}(q_t^d|f_t^{d+1} f_t^d\ q_{t\text{-}1}^d q_t^{d-1}) \quad (7)$$

with f_t^{D+1} and q_t^0 defined as constants.

Shift and reduce probabilities are now defined in terms of finitely recursive FSAs with probability distributions over transition, recursive expansion, and final-state status of states at each hierarchy level. In the HHMM used in this paper, each intermediate state variable is a reduction state variable $f_t^d \in G \cup \{\mathbf{0}, \mathbf{1}\}$ (where G is the set of all nonterminal symbols from the original grammar), representing a reduction to the final syntactic state in G, a horizontal transition to a new awaited category, or a top-down transition to a new active category. Each modeled state variable is a syntactic element ($q_t^d \in G \times G$) with an active and awaited category represented with the slash notation.

The intermediate variable f_t^d is probabilistically determined given a reduction at the stack level below, but is deterministically $\mathbf{0}$ in the case of a non-reduction at the stack level below. [2]

$$\mathsf{P}_{\Theta_{\text{F}}}(f_t^d \mid f_t^{d+1} q_{t\text{-}1}^d q_{t\text{-}1}^{d-1}) \stackrel{\text{def}}{=} \begin{cases} \text{if } f_t^{d+1} \notin G : [f_t^d = \mathbf{0}] \\ \text{if } f_t^{d+1} \in G : \mathsf{P}_{\Theta_{\text{F-Reduce}}}(f_t^d \mid q_{t\text{-}1}^d, q_{t\text{-}1}^{d-1}) \end{cases} \quad (8)$$

where $f^{D+1} \in G$ and $q_t^0 = \mathbf{ROOT}$.

Shift probabilities at each level are defined using level-specific transition $\Theta_{\text{Q-T}}$ and expansion $\Theta_{\text{Q-E}}$ models:

$$\mathsf{P}_{\Theta_{\text{Q}}}(q_t^d \mid f_t^{d+1} f_t^d\ q_{t\text{-}1}^d q_t^{d-1}) \stackrel{\text{def}}{=} \begin{cases} \text{if } f_t^{d+1} \notin G, f_t^d \notin G\text{: } [q_t^d = q_{t\text{-}1}^d] \\ \text{if } f_t^{d+1} \in G, f_t^d \notin G\text{: } \mathsf{P}_{\Theta_{\text{Q-T}}}(q_t^d \mid f_t^{d+1} f_t^d q_{t\text{-}1}^d q_t^{d-1}) \\ \text{if } f_t^{d+1} \in G, f_t^d \in G\text{: } \mathsf{P}_{\Theta_{\text{Q-E}}}(q_t^d \mid q_t^{d-1}) \end{cases} \quad (9)$$

where $f^{D+1} \in G$ and $q_t^0 = \mathbf{ROOT}$. This model is conditioned on final-state switching variables at and immediately below the current HHMM level. If there is no final state immediately below the current level (the first case above), it deterministically copies the current HHMM state forward to the next time step. If there is a final state immediately below the current level (the second case above), it transitions the HHMM state at the current level, according to the distribution $\Theta_{\text{Q-T}}$. And if the state at the current level is final (the third case above), it re-initializes this state given the state at the level above, according to the distribution $\Theta_{\text{Q-E}}$. The overall effect is that higher-level HMMs are allowed to transition only when lower-level HMMs terminate. An HHMM therefore behaves like a probabilistic implementation of a pushdown automaton (or 'shift-reduce' parser) with a finite stack, where the maximum stack depth is equal to the number of levels in the HHMM hierarchy.

All of the probability distributions defined above can be estimated by training on a corpus of right-corner transformed trees, by mapping tree elements onto the random variables in the HHMM and computing conditional probability tables at each random variable. This process is described in more detail in other work (Schuler et al., in press).

3.3 Interruption Point and Word Buffer

This paper expands upon this standard HHMM parsing model by adding two new sub-models to the hidden variables described above, an interruption point (I) variable, and a word buffer (B) . This model is illustrated in Figure 4, which takes Figure 1 as a starting point and adds random variables just mentioned.

Buffers are hypothesized to be used in the human language system to smooth out delivery of speech (Levelt, 1989). In this work, a buffer of that sort is placed between the syntax generating elements and the observed evidence (words). Its role in this model is not to smooth the flow of speech, but to keep a short memory that enables the speaker to conveniently and helpfully restart when a repair is produced. This in turn gives assistance to a listener trying to understand what the speaker is saying, since the listener also has the last few words in memory.

The I variable implements a state machine that keeps track of the repair status at each time point. The domain of this variable is $\{\mathbf{0}, \mathbf{1}, \mathbf{ET}\}$, where $\mathbf{1}$ indicates the first word of an alteration, $\mathbf{ET}$ indicates editing terms in between reparandum and alteration, and $\mathbf{0}$ indicating no repair.[3]

[2] Here $[\cdot]$ is an indicator function: $[\phi] = 1$ if ϕ is true, 0 otherwise.

[3] Actually, $\mathbf{0}$ can occur during an alteration, but in those cases that fact is indicated by the state of the buffer.

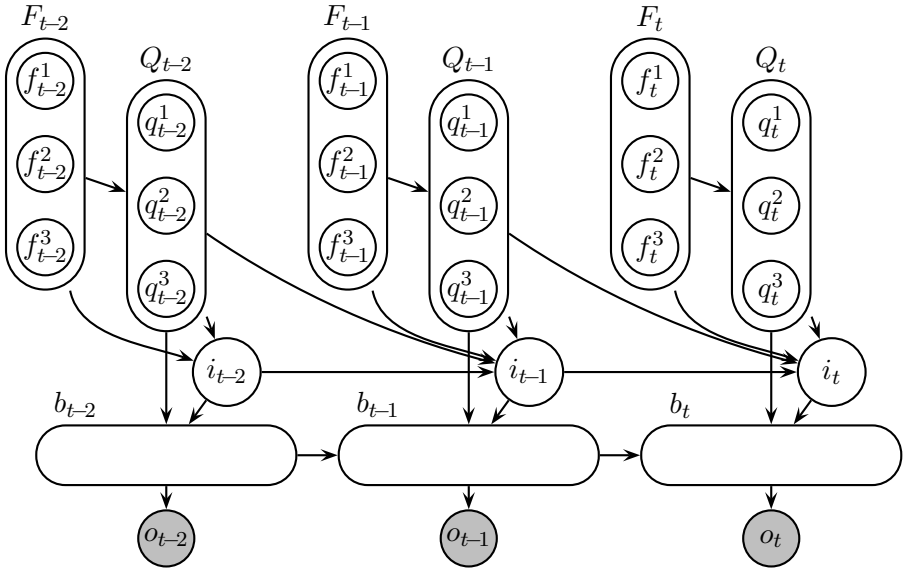


Figure 4: Extended HHMM parsing model with variables for interruption points (I) and a modeled word buffer (B). Arrows within and between complex hidden variables F and Q have been removed for clarity.

The value of I is deterministically constrained in this work by its inputs, but it can be conceived as a conditional probability $\mathsf{P}(i_t \mid i_{t-1}, q_t, q_{t-1}, r_t)$ to allow footholds for future research.[4] While depending formally on many values, in practice its dependencies are highly context-dependent and constrained:

$$\mathsf{P}(i_t \mid i_{t-1}, q_t, q_{t-1}, q_t) \stackrel{\text{def}}{=}$$

$$\begin{cases} \text{if } i_{t-1} = \mathbf{1} & : [i_t = \mathbf{0}] \\ \text{if } i_{t-1} = \mathbf{ET} \wedge (INTJ \vee PRN) \in q_t & : [i_t = \mathbf{ET}] \\ \text{if } i_{t-1} = \mathbf{ET} & : [i_t = \mathbf{1}] \\ \text{if } i_{t-1} = \mathbf{0} \wedge EDITED \in (q_{t-1} \cup f_t) & \\ \quad \wedge (INTJ \vee PRN) \in q_t & : [i_t = \mathbf{ET}] \\ \text{if } i_{t-1} = \mathbf{0} \wedge EDITED \in (q_{t-1} \cup f_t) & : [i_t = \mathbf{1}] \\ \text{if } i_{t-1} = \mathbf{0} & : [i_t = \mathbf{0}] \end{cases}$$

These conditions are meant to be evaluated in a short-circuiting fashion, i.e., the first condition which is true starting from the top is applied. The default (last) case is most common, going from non-repair to non-repair state. When the syntax generated something with the category EDITED at the last time step (as evidenced by either the modeled state variable q_{t-1} or the reduction state variable f_t depending on the length of the reparandum), the interruption point variable is triggered to change, either to $\mathbf{ET}$ if an interjection (INTJ) or parenthetical (PRN) followed, otherwise to $\mathbf{1}$ for the first word of an alteration. The $\mathbf{ET}$ state continues as long as the syntax at the current level is generating something containing INTJ or PRN.

The random variable for the word buffer is more complex, containing at each time step t an integer index for keeping track of a current position in the buffer ($c_t \in \langle 0, 1, \ldots, n-1 \rangle$ for buffer size n), and an array of several recently generated words ($\vec{w}_t$). This can be represented as the following conditional probability:

$$\mathsf{P}(b_t \mid b_{t-1}, i_t, q_t) = \mathsf{P}(c_t \mid c_{t-1}, i_t) \cdot \mathsf{P}(\vec{w}_t \mid \vec{w}_{t-1}, c_t) \qquad (10)$$

The operation of the buffer is governed by four cases:

Case 1: During normal operation (i.e. for fluent speech), the interruption point variable is $\mathbf{0}$ and at the previous time step the buffer index points at the end of the buffer ($i_t = \mathbf{0} \wedge c_{t-1} = n-1$). In this simple case, the buffer pointer remains pointing at the end position in the buffer ($c_t = n-1$), and the last $n-1$ items in the buffer are deterministically copied backwards one position. A new word is generated probabilistically to occupy the last position in the buffer (where c_t is pointing). This probability is estimated empirically using the same model used in a standard HHMM to generate words, by conditioning the word on the deepest non-empty q_t value in the stack.

Case 2: When an editing term is being generated, ($i_t = \mathbf{ET}$), the buffer is not in use. Practi-

[4] Most obviously, this variable could be made prior to its conditions to be their cause, if a suitable model for the causation of interruption points was designed using prosodic cues. For this work, it is simply an intermediary that is not strictly necessary but makes the model design more intuitive.

cally, this means that the value of the index c and all w^j are just copied over from time $t-1$ to time t. This makes sense psycholinguistically, because a buffer used to smooth speech rates would by definition not be used when speech is interrupted by a repair. It also makes sense from a purely engineering point of view, since words used as editing terms are usually stock phrases and filled pauses that are not likely to have much predictive value for the alteration, and are thus not worth keeping in the buffer. The probability of the actual observed word is modeled the same way word probabilities are modeled in a standard HHMM, conditioned on the deepest non-empty q_t value, and ignoring the buffer.

Case 3: The alteration case applies to the first word after the reparandum and optional editing terms ($i_t = \mathbf{1}$). In this case, the index c_t for the current position of the buffer is obtained by subtracting a number of words to replace, with that number drawn from a prior distribution. This distribution is based on the function $f(k) = 1.22 \cdot 0.45^k$. This function was taken from Shriberg (1996), where it was estimated based on several different training corpora, and provided a remarkable fit to all of them. Since this model uses a fixed size buffer, the values are precomputed and renormalized to form a probability distribution. With a buffer size of only $n = 4$, approximately 96% of the probability mass of the original function is accounted for.

After the indices are computed, the buffer at position c_t is given a word value. The model first decides whether to substitute or copy the previous word over. The probability governing this decision is also determined empirically, by computing how often the first word in a alteration in the Switchboard training set is a copy of the first word it is meant to replace. If the copy operation is selected, the word is added to the buffer without further diluting its probability. If, however, the substitution operation was selected, the word is added to the buffer with probability distributed across all possible words.

Case 4: The final case to account for is alterations of length greater than one ($i_t = \mathbf{0} \wedge c_{t-1} \neq n-1$). This occurs when the current index was moved back more than one position, and so even though i is set to $\mathbf{0}$, the current index into the buffer is not pointing at the end. In this case, again the index c_t is selected according to a prior probability distribution. The value selected from the distribution corresponds to different actions that may be selected when retracing the words in the reparandum to generate the alteration.

The first option is that the current index remains in place, which corresponds to an *insertion* operation, where the alteration is given an extra word relative to the reparandum at its current position. Following an insertion, a new word is generated and placed in the buffer at the current index, with probability conditioned on the syntax at the most recent time step. The second option is to continue the alignment, moving the current index forward one position in the buffer, and then either performing a *substitution* or *copy* operation in alignment with a word from the alteration. Word probabilities for the copy and substitution operations are generated in the same way as for the first word of an alteration. Finally, the current index may skip forward more than one value, performing a *deletion* operation. Deletion skips over words in the reparandum that do not correspond to words in the alteration. After the deletion moves the current index pointer forward, a word is again either copied or substituted against the newly aligned word.

The prior probability distributions over alignment operations is estimated from data in the Switchboard in a similar manner to Johnson and Charniak (2004). Briefly, using the disfluency-annotated section of the Switchboard corpus (.dps files), a list of reparanda and alterations corresponding to one another are compiled. For each pair, the minimal cost alignment is computed, where a copy operation has cost 0, substitution has cost 4, and deletion and insertion each have cost 7. Using these alignments, probabilities are computed using relative frequency counts for both the first word of an alteration, and for subsequent operations. Copy and substitution are the most frequent operations (copying gives information about the repair itself, while substitution can correct the reason for the error), insertion is somewhat less frequent (presumably for specifying further information), and deletion is relatively rare (usually a repair is not made to remove information).

4 Evaluation

This model was evaluated on the Switchboard corpus (Godfrey et al., 1992) of conversational telephone speech between two human interlocu-

System	*Precision*	*Recall*	*F-Score*
Plain CYK	18.01	17.73	17.87
Hale et al. CYK	40.90	35.41	37.96
Hale et al. Lex.	n/a	n/a	*70.0*
TAG	*82.0*	*77.8*	*79.7*
Plain HHMM	43.90	47.36	45.57
HHMM-Back	**44.12**	**57.49**	**49.93**
HHMM-Retrace	**48.82**	**59.41**	**53.59**

Table 1: Table of results of edit-finding accuracy. Italics indicate reported, rather than reproduced, results.

System Configuration	*Parseval-F*	*Edited-F*
Plain CYK	71.03	17.9
Hale et al. CYK	68.47	37.96
Hale et al. Lex.	*80.16*	*70.0*
Plain HHMM	74.23	45.57
HHMM-Back	**74.58**	**49.93**
HHMM-Retrace	**74.23**	**53.59**

Table 2: Table of parsing results.

tors. The input to this system is the gold standard word transcriptions, segmented into individual utterances. The standard train/test breakdown was used, with sections 2 and 3 used for training, and subsections 0 and 1 of section 4 used for testing. Several held-out sentences from the end of section 4 were used during development.

For training, the data set was first standardized by removing punctuation, empty categories, typos, all categories representing repair structure, and partial words – anything that would be difficult or impossible to obtain reliably with a speech recognizer.

The two metrics used here are the standard Parseval F-measure, and Edit-finding F. The first takes the F-score of labeled precision and recall of the non-terminals in a hypothesized tree relative to the gold standard tree. The second measure marks words in the gold standard as edited if they are dominated by a node labeled EDITED, and measures the F-score of the hypothesized edited words relative to the gold standard.

Results are shown in Tables 1 and 2. Table 1 shows detailed results on edited word finding, with two test systems and several related approaches.

The first two lines show results from a reimplementation of Hale et al. parsers. In both those cases, gold standard part-of-speech (POS) tags were supplied to the parser. The following two lines are reported results of a lexicalized parser from Hale et al. and the TAG system of Johnson and Charniak. The final three lines are evaluations of HHMM systems. The first is an implementation of Miller and Schuler, run without gold standard POS tags as input. The second HHMM result is a system much like that described in this paper, but designed to approximate the best result that can come from simply trying to match the first word of an alteration with a recent word. Levelt (1989) notes that in over 90% of repairs, the first word of the alteration is either identical or a member of the same category as the first word of the reparandum, and this clue is enough for listeners to understand what the alteration is meant to replace. This implementation keeps the I variable to model repair state, but rather than a modeled buffer being part of the hidden state, it keeps an observed buffer that simply tracks the last n words seen ($n = 4$ in this experiment). This buffer is used only to generate the first word of a repair, and only when the syntactic state allows the word. Finally, the system described in Section 3 is shown on the final line.

Table 2 shows overall parsing accuracy results, with the same set of systems, with the exception of the TAG system which did not report parsing results.

5 Discussion and Conclusion

These results first show that the main contribution of this paper, a model for a buffer of recent words which influences speech repairs, results in drastic improvements in the ability of an HHMM system to discover edited words. This model does this in a single pass through the observed words, incrementally forming hypotheses about the state of the syntactic process as well as the state of repair, just as humans must recognize spontaneous speech.

Another interesting result is the relative effectiveness of a buffer that is not modeled, but rather just a collection of words used to condition the first words of repair ('HHMM-Back'). While this result is superior to the plain HHMM system, it still falls well short of the retracing model using a modeled buffer. This suggests that, though one word is sufficient to align a reparandum and alteration when the existence of a repair is given, more information is often necessary when the task is not just alignment of repair but also detection of re-

pair. A model that takes into account information sources that identify the existence of repair, such as prosodic cues (Hale et al., 2006; Lickley, 1996), may thus result in improved performance for the simpler unmodeled buffer.

These results also confirm that parsing spontaneous speech with an HHMM can be far superior to a CKY parser, even when the CKY parser is given the advantage of correct POS tags as input. Second, even the baseline HHMM system also improves over the CYK parser in finding edited words, again without the advantage of correct POS tags as input.

In conclusion, the model described here uses a buffer inspired by the phonological loop used in the human auditory system to keep a short memory of recent input. This model, when used to assist in the detection and correction of repair, results in a large increase in accuracy in detection of repair over other most basic parsing systems. This system does not reach the performance levels of lexicalized parsers, nor multi-pass classification systems. Future work will explore ways to apply additional features of these systems or other sources of information to account for the remainder of the performance gap.

References

Alan Baddeley, Susan Gathercole, and Costanza Papagno. 1998. The phonological loop as a language learning device. *Psychological Review*, 105(1):158–173, January.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 118–126.

Mark G. Core and Lenhart K. Schubert. 1999. A syntactic framework for speech repairs and other disruptions. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL 99)*.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proc. ICASSP*, pages 517–520.

John Hale, Izhak Shafran, Lisa Yung, Bonnie Dorr, Mary Harper, Anna Krasnyanskaya, Matthew Lease, Yang Liu, Brian Roark, Matthew Snover, and Robin Stewart. 2006. PCFGs with syntactic and prosodic indicators of speech repairs. In *Proceedings of the 45th Annual Conference of the Association for Computational Linguistics (COLING-ACL)*.

Peter A. Heeman and James F. Allen. 1999. Speech repairs, intonational phrases, and discourse markers: Modeling speakers' utterances in spoken dialogue. *Computational Linguistics*, 25:527–571.

Mark Johnson and Eugene Charniak. 2004. A tag-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*, pages 33–39, Barcelona, Spain.

Mark Johnson. 1998a. Finite state approximation of constraint-based grammars using left-corner grammar transforms. In *Proceedings of COLING/ACL*, pages 619–623.

Mark Johnson. 1998b. PCFG models of linguistic tree representation. *Computational Linguistics*, 24:613–632.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430.

Willem J.M. Levelt. 1989. *Speaking: From Intention to Articulation*. MIT Press.

R. J. Lickley. 1996. Juncture cues to disfluency. In *Proceedings of The Fourth International Conference on Spoken Language Processing (ICSLP '96)*, pages 2478–2481.

Tim Miller and William Schuler. 2008. A syntactic time-series model for parsing fluent and disfluent speech. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08)*.

Kevin P. Murphy and Mark A. Paskin. 2001. Linear time inference in hierarchical HMMs. In *Proc. NIPS*, pages 833–840.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. in press. Broad-coverage incremental parsing using human-like memory constraints. *Computational Linguistics*.

William Schuler. 2009. Parsing with a bounded stack using a model-based right-corner transform. In *Proceedings of the North American Association for Computational Linguistics (NAACL '09)*, Boulder, Colorado.

Elizabeth Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.D. thesis, University of California at Berkeley.

Elizabeth Shriberg. 1996. Disfluencies in Switchboard. In *Proceedings of International Conference on Spoken Language Processing*.

Less is More: Significance-Based N-gram Selection for Smaller, Better Language Models

Robert C. Moore **Chris Quirk**
Microsoft Research
Redmond, WA 98052, USA
{bobmoore,chrisq}@microsoft.com

Abstract

The recent availability of large corpora for training N-gram language models has shown the utility of models of higher order than just trigrams. In this paper, we investigate methods to control the increase in model size resulting from applying standard methods at higher orders. We introduce significance-based N-gram selection, which not only reduces model size, but also improves perplexity for several smoothing methods, including Katz backoff and absolute discounting. We also show that, when combined with a new smoothing method and a novel variant of weighted-difference pruning, our selection method performs better in the trade-off between model size and perplexity than the best pruning method we found for modified Kneser-Ney smoothing.

1 Introduction

Statistical language models are potentially useful for any language technology task that produces natural-language text as a final (or intermediate) output. In particular, they are extensively used in speech recognition and machine translation. Despite the criticism that they ignore the structure of natural language, simple N-gram models, which estimate the probability of each word in a text string based on the $N-1$ preceding words, remain the most widely-used type of model.

Until the late 1990s, N-gram language models of order higher than trigrams were seldom used. This was due, at least in part, to the fact the amounts of training data available did not produce significantly better results from higher-order models. Since that time, however, increasingly large amounts of language model training data have become available ranging from approximately one billion words (the Gigaword corpora from the Linguistic Data Consortium) to trillions of words (Brants et al., 2007). With these larger resources, the use of language models based on 5-grams to 7-grams is becoming increasingly common.

The problem we address here is that, even when relatively modest amounts of training data are used, high-order N-gram language models estimated by standard techniques can be impractically large. Hence, we investigate ways of building high-order N-gram language models without dramatically increasing model size. This is, of course, the same goal behind much previous work on language model pruning, including that of Seymore and Rosenfeld (1996), Stolcke (1998), and Goodman and Gao (2000). We take a novel approach, however, which we refer to as significance-based N-gram selection. We reject a higher-order estimate of the probability of a particular word in a particular context whenever the distribution of observations for the higher-order estimate provides no evidence that the higher-order estimate is better than our backoff estimate.

Perhaps our most surprising result is that significance-based N-gram selection not only reduces language model size, but it also improves perplexity when applied to a number of widely-used smoothing methods, including Katz backoff and several variants of absolute discounting.[1] In contrast, experiments applying previous pruning methods to Katz backoff (Seymore and Rosenfeld, 1996; Stolcke, 1998) and absolute discounting (Goodman and Gao, 2000) always found the lowest perplexity model to be the unpruned model.

We tested significance-based selection on only one smoothing method without obtaining improved perplexity: modified Kneser-Ney (KN)

[1] For most of the standard smoothing methods mentioned here, we refer the reader to the excellent comparative study of smoothing methods by Chen and Goodman (1998). References to the original sources may be found there.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 746–755,
Singapore, 6-7 August 2009.

smoothing (Chen and Goodman, 1998). This is unfortunate, because modified KN smoothing generally seems to have the lowest perplexity of any known smoothing method for N-gram language models; in our tests it had a lower perplexity than any of the other models, with or without significance-based N-gram selection. However, when we compared modified KN smoothing to our best results applying N-gram selection to other smoothing methods for multiple N-gram orders, two of our models outperformed modified KN in terms of perplexity for a given model size.

Of course, the trade-off between perplexity and model size for modified KN can also be improved by pruning. So, in a final set of experiments we found the best combinations we could for pruned modified KN models, and we did the same for our best model using significance-based selection. The best pruning method for the latter turned out to be a novel modification of weighted-difference pruning (Seymore and Rosenfeld, 1996) that was especially convenient to compute given our method for performing significance-based N-gram selection. The final result is that our best model using significance-based selection and modified weighted difference pruning always had a better size/perplexity trade-off than pruned modified KN, with up to about 8% perplexity reduction for a given model size.

2 Significance-Based N-gram Selection

The idea of using a statistical test to decide whether to use a higher- or lower-order estimate of an N-gram probablity is not new. It was perhaps first proposed by Ron, et al. (1996), who suggested using a threshold on relative entropy (Kullback-Liebler divergence) as an appropriate test to decide whether to extend the context used to predict the next token in a sequence. Stolcke (1998) used the same metric in his work on language model pruning, and he also pointed out that weighted difference pruning is, in fact, an approximation of relative entropy pruning. However, while relative entropy pruning is based on a statistical test, it is not a *significance* test. The difference in probability represented by a certain relative entropy value can be statistically significant when measured on a large corpus, but not significant when measured on a small corpus.

The primary test we use to choose between higher- or lower-order estimates of an N-gram probablity is inspired by an insight of Jedynak and Khudanpur (2005). They note that, given a set of y observations of a multinomial distribution, the observed counts will have the highest probabilty of any possible set of y observations for the maximum likelihood estimate (MLE) model derived from the relative frequencies of those observations. In general, however, the MLE model will not be the only model for which this set of observations is the most probable set of y observations. Jedynak and Khudanpur call the set of such models the maximum likelihood set (MLS) for the observations.

Jedynak and Khudanpur argue that the observations alone do not support choosing the MLE over other members of the MLS. The MLE may assign the observations a higher probability than other members of the MLS, but that may be an accident of what outcomes are possible given the number of observations. If we flip a coin 9 times and get 5 heads, is there any reason to believe that the probability of heads is closer to the MLE $5/9$ than it is to $5/10$? No, because $5/9$ is as close as we can come to $5/10$, given 9 observations.

We apply this insight to the problem of N-gram selection as follows: For each word w_n in a context $w_1...w_{n-1}$ with a backoff estimate for the probability of that word in that context $\beta\, p(w_n|w_2...w_{n-1})$,[2] we do not include an explicit estimate of $p(w_n|w_1...w_{n-1})$ in our model, if the backoff estimate is within the MLS of the counts for $w_1...w_n$ and $w_1...w_{n-1}$.

This requires finding the MLS of a set of observations only for binomial distributions (rather than the general multinomial distributions studied by Jedynak and Khudanpur), which has a very simple solution:

$$MLS(x,y) = \left\{ p \;\middle|\; \frac{x}{y+1} \leq p \leq \frac{x+1}{y+1} \right\}$$

where x is the count for $w_1...w_n$, y is the count for $w_1...w_{n-1}$, and p is a possible backoff probabilty estimate for $p(w_n|w_1...w_{n-1})$. In this case, the MLS is the set of binomial distributions that have x successes as their mode given y trials, which is well-known to be specified by this formula.

We describe this method as "significance-based" because we can consider our criterion as a significance test in which we take the backoff

[2] $p(w_n|w_2...w_{n-1})$ being the next lower-order estimate, and β being the backoff weight for the context $w_1...w_{n-1}$.

probability estimate as the null hypothesis for the estimate in the higher-order model, and we set the rejection threshold to the lowest possible value; we reject the null hypothesis (the backoff probability) if there are *any* outcomes for the given number of trials that are more likely, according to the null hypothesis, than the one we observed.

We make a few refinements to this basic idea. First, we never add an explicit higher-order estimate to our model, if the next lower-order estimate is not explicitly stored in the model. This enables us to keep only the next lower-order model available while performing N-gram selection.

Next, we observe that in some cases the higher-order estimate for $p(w_n|w_1...w_{n-1})$ may not fall within the MLS for the observed counts, due to smoothing. In this case, we prefer the backoff probability estimate if it lies within the MLS or between the smoothed higher-order estimate and the MLS. Otherwise, we would reject the backoff estimate for being outside the MLS, only to replace it with a higher-order estimate even farther outside the MLS.

Finally, we note that the backoff probability estimate for an N-gram not observed in the training data sometimes falls outside the corresponding MLS, which in the 0-count case simplifies to

$$MLS(0, y) = \left\{ p \;\middle|\; 0 \leq p \leq \frac{1}{y+1} \right\}$$

When this happens, we include an explicit higher-order estimate $p(w_n|w_1...w_{n-1}) = 1/(y+1)$, which is the upper limit of the MLS. This is similar to Rosenfeld and Huang's (1993) "confidence interval capping" method for reducing unreasonably high backoff estimates for unobserved N-grams.

In order to apply this treatment of 0-count N-grams, we sort the explicitly-stored N-grams for each backoff context by decreasing probability. For each higher-order context, to find the 0-count N-grams subject to the $1/(y+1)$ limit, we traverse the sorted list of explicitly-stored N-grams for its backoff context. When we encounter an N-gram whose extension to the higher-order context was not observed in the training data, we give it an explicit probability of $1/(y+1)$, if its weighted backoff probability is greater than that. We stop the traversal as soon as we encounter an N-gram for the backoff context that has a weighted backoff probability less than or equal to $1/(y+1)$, which in practice means we actually examine only a small number of backoff probabilities for each context.

3 Finding Backoff Weights by Iterative Search

The approach described above is very attractive from a theoretical perspective, but it has one practical complication. To decide which N-grams for each context to explicitly include in the higher-order model, we need to know the backoff weight for the context, but we cannot compute the backoff weight until we know exactly which higher-order N-grams are included in the model.

We address this problem by iteratively solving for a backoff weight that yields a normalized probability distribution. For each context, we guess an initial value for the backoff weight and keep track of the sum of the probabilites resulting from applying our N-gram selection method with that backoff weight. If the sum is greater than 1.0, by more than a convergence threshold, we reduce the estimated backoff weight and iterate. If the sum is less than 1.0, by more than the threshold, we increase the estimated weight and iterate.

It is easy to see that, for all standard smoothing methods, the function from backoff weights to probability sums is piece-wise linear. Within a region where no decision changes about which N-grams to include in the model, the probability sum is a linear function of the backoff weight. At values of the backoff weight where the set of selected N-grams changes, the function can be discontinous. With a little more effort, one can see that the linear segments overlap with respect to the probability sum in such a way that there will always be one or more values of the backoff weight that make the probability sum equal 1.0, with one specific exception.

The exception arises because of the capping of backoff probabilites for unobserved N-grams. It is possible for there to be a context for which all observed N-grams are included in the higher-order model, the probabilities for all unobserved N-grams are either capped at $1/(y+1)$ or effectively 0 due to arithmetic underflow, and the probability sum is less than 1.0. For some smoothing methods, the probability sum cannot be increased in this situation by increasing the backoff weight. We check for this situation, and if it arises, we increase the cap on the 0-count probability just enough to make the probability sum equal 1.0.

That exception aside, we iteratively find backoff weights as follows: For an initial estimate of the backoff weight for a context, we compute

what the backoff weight would be for the base smoothing method without N-gram selection. If that value is less than 1.0, we use it as our initial estimate, otherwise we use 1.0, which annecdotally seems to produce better models than initial estimates greater than 1.0, in situations where there are multiple solutions. If the first iteration of N-gram selection produces a probability sum less than 1.0, we repeatedly double the estimated backoff weight until we obtain a sum greater than or equal to 1.0, or we encounter the special situation previously described. If the initial probability sum is greater than 1.0, we repeatedly halve the estimated backoff weight until we obtain a sum less than or equal to 1.0.

Once we have values for the backoff weight that produce probability sums on both sides of 1.0, we have a solution bracketed, and we can use standard numerical search techniques to find that solution. At every subsequent iteration, we try a value for the backoff weight between the largest value we have tried that produces a sum less than 1.0 and the smallest value we have tried that produces a sum greater than 1.0. We stop when the difference between these values of the backoff weight is less than a convergence threshold.

We use a combination of simple techniques to choose the next value of the backoff weight to try. The primary technique we use is called the "false position method", which basically solves the linear equation defined by the two current bracketing values and corresponding probability sums. The advantage of this method is that, if our bracketing points lie on the same linear segment of our function, we obtain a solution in one step. The disadvantage of the method is that it sometimes approaches the solution by a long sequence of tiny steps from the same side.

We try to detect the latter situation by keeping track of the number of consecutive iterations that make a step in the same direction. If this number reaches 10, we take the next step by the bisection method, which simply tries the value of the backoff weight halfway between our two current bracketing values. In practice, this combined search method works very well, taking an average of less than four iterations per backoff weight.

4 Modified Weighted-Difference Pruning

While the N-gram selection method described above considerably reduces the number of parameters in a high-order language model, we may wish to reduce language model size even more. The concept of significance-based N-gram selection to produce smaller models could be extended by relaxing our criterion for using backoff distributions in place of explicit higher-order probability estimates, but true significance tests at more relaxed thresholds that are accurate for small counts are expensive to compute; so we resort to more conventional language model pruning methods.

In our experiments, we tried four methods for additional pruning: simple count cutoffs, relative entropy pruning (REP) (Stolcke, 1998), and two modified versions of Seymore and Rosenfeld's (1996) weighted-difference pruning (WDP). In the notation we have been using, Seymore and Rosenfeld's WDP criterion for using a backoff estimate, in place of an explicit higher-order estimate, is that the quantity

$$K \times \left(\begin{array}{l} \log(p(w_n|w_1...w_{n-1})) - \\ \log(\beta_u \, p(w_n|w_2...w_{n-1})) \end{array} \right)$$

be less than a pruning threshold, where K is the Good-Turing-discounted training set count for $w_1...w_n$, and β_u is the backoff weight for the unpruned model.

The first of our modified version of WDP uses the following quantity instead:

$$p(w_1...w_n) \times \left| \begin{array}{l} \log(p(w_n|w_1...w_{n-1})) - \\ \log(\beta_p \, p(w_n|w_2...w_{n-1})) \end{array} \right|$$

where $p(w_1...w_n)$ is an estimate of the probability of $w_1...w_n$ and β_p is the backoff weight for the pruned model.

We make three modifications to WDP in this formula. First, we follow a suggestion of Stolcke (1998) by replacing the discounted training set count K of $w_1...w_n$ with an estimate the joint probability of $w_1...w_n$, computed by chaining the explicit probability estimates, according to our model, for all N-gram lengths up to n.

The second modification to WDP is that we use the absolute value of the difference of the log probabilities. By using the signed difference of the log probabilities, Seymore and Rosenfeld will always prune a higher-order probability estimate if it is less than the backoff estimate. But the backoff estimate may well be too high. Using the absolute value of the difference avoids this problem.

$$p(w_n|w_1 \dots w_{n-1}) = \begin{cases} \alpha_{w_1 \dots w_{n-1}} \frac{C(w_1 \dots w_n) - D_{n,C(w_1 \dots w_n)}}{C(w_1 \dots w_{n-1})} & \\ \quad + \beta_{w_1 \dots w_{n-1}} p(w_n|w_2 \dots w_{n-1}) & \text{if } C(w_1 \dots w_n) > 0 \\ \\ \gamma_{w_1 \dots w_{n-1}} p(w_n|w_2 \dots w_{n-1}) & \text{if } C(w_1 \dots w_n) = 0 \end{cases}$$

$$\beta_{w_1 \dots w_{n-1}} = \delta \frac{|\{w' | C(w_1 \dots w_{n-1} w') > 0\}|}{C(w_1 \dots w_{n-1})}$$

$$\alpha_{w_1 \dots w_{n-1}} = 1 - \beta_{w_1 \dots w_{n-1}}$$

Figure 1: New language model smoothing method

The final modification is that we compute the difference in log probability with respect to the backoff weight for the pruned model rather than the unpruned model, which we are able to do by performing the pruning inside our iterative search for the value of the backoff weight. We do this because, if the backoff weight is changed significantly by pruning, backoff estimates that meet the pruning criterion with the old backoff weight may no longer meet the criterion with the new backoff weight, and vice versa. Since the new backoff weight is the one that will be used in the pruned model, that seems to be the one that should be used to make pruning decisions.

Our second variant of modified WDP is like the first, but it estimates $p(w_1 ... w_n)$ simply by dividing Seymore and Rosenfeld's discounted N-gram count K by the total number of highest-order N-grams in the training corpus. This is equivalent to smoothing only the highest-order conditional N-gram model in estimating $p(w_1 ... w_n)$, estimating all the lower-order probabilities in the chain by the corresponding MLE model. We refer to this joint probability estimate as "partially-smoothed", and the one suggested by Stolcke as "fully-smoothed".

5 Evaluation

We carried out three sets of evaluations to test the new techniques described above. First we compared the perplexity of full models and models reduced by significance-based N-gram selection for seven language model smoothing methods. For the best three results in that comparison, we looked at the trade-off between perplexity and model size over a range of N-gram orders. Finally, we tried various pruning methods to further reduce model size, and then compared the best result we obtained using previous techniques with the best result we obtained using our new techniques.

5.1 Data and Base Smoothing Methods

For training, parameter optimzation, and test data we used English text from the WMT-06 Europarl corpus (Koehn and Monz, 2006). We trained on the designated 1,003,349 sentences (27,493,499 words) of English language model training data, and used 2000 sentences each for testing and parameter optimization, from the English half of the English-French dev and devtest data sets.

We conducted our experiments on seven language model smoothing methods. Five of these are well-known: (1) interpolated absolute discounting with one discount per N-gram length, estimated according to the formula derived by Ney et al. (1994); (2) Katz backoff with Good-Turing discounts for N-grams occurring 5 times or less; (3) backoff absolute discounting with Ney et al. formula discounts; (4) backoff absolute discounting with one discount used for all N-gram lengths, optimized on held-out data; (5) modified interpolated Kneser-Ney smoothing with three discounts per N-gram length, estimated according to the formulas suggested by Chen and Goodman (1998).

We also experimented with two variants of a new smoothing method that we have recently developed. Full details of the new method are given elsewhere (Moore and Quirk, 2009), but since it is not well-known, we summarize the method here. Smoothed N-gram probabilities are defined by the formulas shown in Figure 1, for all n such that $N \geq n \geq 2$,[3] where N is the greatest N-gram length used in the model. The novelty of this model is that, while it is an interpolated model, the interpolation weights β for the lower-order model

[3]For $n = 2$, we take the expression $p(w_n|w_2 \dots w_{n-1})$ to denote a unigram probability estimate $p(w_2)$.

	Method	base PP	select PP	percent change
1	interp-AD-fix	62.6	61.6	-1.6
2	Katz backoff	59.8	56.1	-7.9
3	backoff-AD-fix	59.9	54.3	-9.3
4	backoff-AD-opt	58.8	54.4	-7.5
5	KN-mod-fix	51.6	54.6	+5.8
6	new-fix	56.1	52.1	-7.1
7	new-opt	53.7	52.0	-3.3

Table 1: Perplexity results for N-gram selection

are not constrained to match the backoff weights γ for the lower-order model. This allows the interpolation weights to be set independently of the discounts D, with the backoff weights being adjusted to normalize the resulting distributions.

The motivation for this is to let the D parameters correct for potential overestimation of the probabilities for observed N-grams, while the δ parameter (which determines the α and β interpolation parameters) somewhat independently corrects for quantization errors caused by the fact that only certain probabilities can be derived from integer observed counts, even after discounting. δ is interpretable as the estimated mean quantization error for each distinct count for a given context.

We tested two variants of the new method, (6) one in which the D parameters and the δ parameter are set by fixed criteria, and (7) one in which a single value for all D parameters and the value of the δ parameter are optimized on held-out data. For the fixed value of δ, we assume that, since the distance between possible N-gram counts, after discounting, is approximately 1.0, their mean quantization error would be approximately 0.5. For the fixed discount parameters, we use three values for each N-gram length: D_1 for N-grams whose count is 1, D_2 for N-grams whose count is 2, and D_3 for N-grams whose count is 3 or more. We set these values to be the discounts for 1-counts, 2-counts, and 3-counts estimated by the Good-Turing method. This yields the formula

$$D_r = r - (r+1)\frac{N_{r+1}}{N_r},$$

for $1 \leq r \leq 3$, where N_r is the number of distinct N-grams of the length in question occuring r times in the training set.

In all experiments, the unigram language model is an un-smoothed, closed-vocabulary MLE model. We use this unigram model, because there is no simple, principled way of assigning probabilities to individual out-of-vocabulary (OOV) words. The only principled solution to this problem that we are aware of is to use a character-based model, but this seems overly complicated for something that is orthogonal to the main points of this study, and of minor practical importance. Since we make no provision for OOV words in the models, OOV words are also omitted from all perplexity measurements. Thus, the perplexity numbers are systematically lower than they would be if OOVs were taken into account, but they are all comparable in this regard.

5.2 Results for Significance-Based N-gram Selection

Table 1 shows the minimum perplexity (with respect to N-gram order) of language models up to 7-grams for each of the seven smoothing methods discussed above, with and without significance-based N-gram selection. N-gram selection improved the perplexity of all models, except for modified KN. The lowest overall perplexity remains that of the base modified KN method, but with N-gram selection, the two variants of the new smoothing method come very close to it.

If we cared only about perplexity, that would be the end of the story, but we also care about language model size. The results in Table 1 were obtained on models estimated using just the counts needed to cover the parameter optimization and test sets; so to accurately measure model size, we trained full language models using base modifed KN, and the two variants of the new method with N-gram selection. The resulting sizes of the models represented in backoff form (in terms of total number of probability and backoff parameters) are shown in Figure 2 as function of N-gram length, from trigrams up to 7-grams for KN and up to 10-grams for the two new models. We see that beyond 4-grams the model sizes diverge dramatically, with the new models incorporating N-gram selection leveling off, but the modified KN model (or any standard model) continuing to grow in size, apparently linearly in the N-gram order.

In Figure 3, we show the relationship between perplexity and model size for the same three models, varying N-gram order. We see that between about 20 million and 45 million parameters, both of the new models incorporating significance-

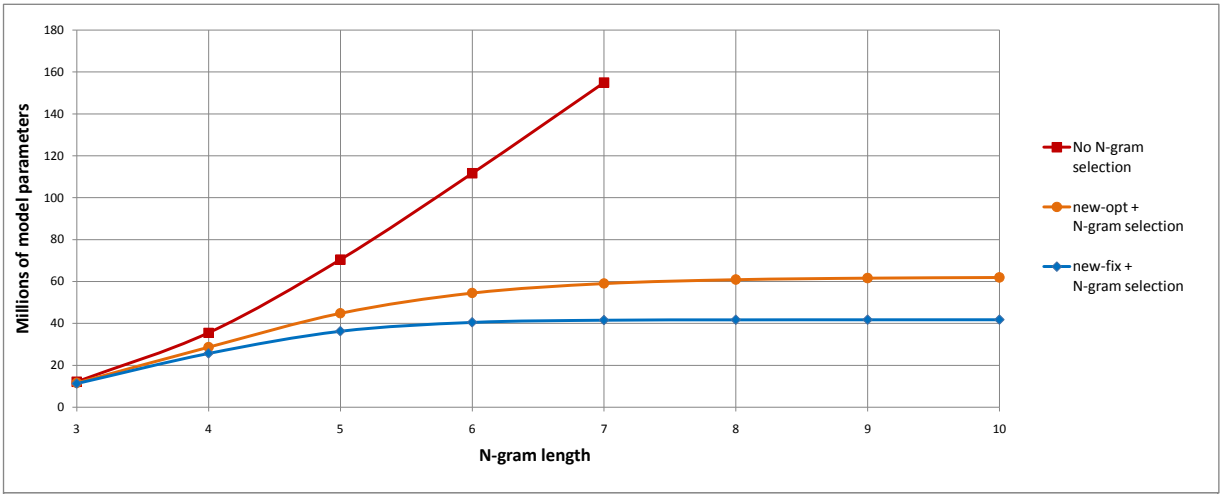


Figure 2: Model size vs. N-gram length

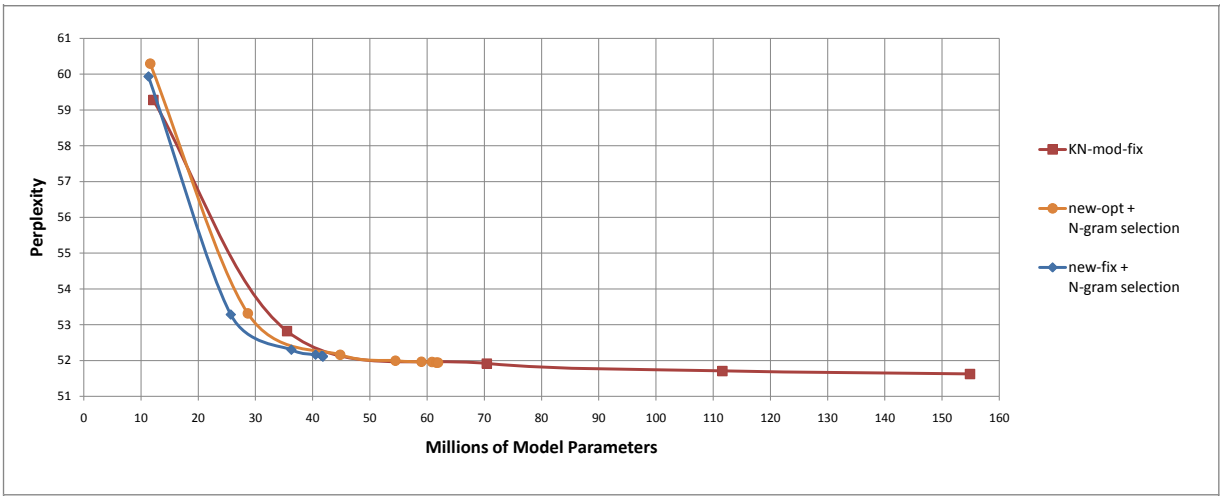


Figure 3: Perplexity vs. model size

based N-gram selection seem to outperform modified KN, and that the best of the three is, in fact, the new model with fixed parameter values.

5.3 Results for Additional Pruning

We further tested modified KN smoothing, and our new smoothing method with fixed parameter values and significance-based N-gram selection, with additional pruning. We compared several pruning methods on trigram models: count cutoffs, REP,[4] and our two modified versions of WDP.

Figure 4 shows the resulting combinations of perplexity and model size for REP and modified WDP at various pruning thresholds, and for count cutoffs of 1, 2, and 3 for both bigrams and trigrams ($n > 1$) and for trigrams only ($n > 2$), applied to our new smoothing method with fixed parameter values, together with significance-based N-gram selection. Overall, modified WDP with fully-smoothed joint probability estimates performs the best. It is has lower perplexity than count cutoffs at all model sizes tested, and is about equal to REP at very severe pruning levels and superior to REP with less pruning. Modified WDP with fully-smoothed joint probabilities is about equal to modified WDP with partially-smoothed joint probabilities at the highest and lowest pruning levels tested, but superior in between.

Figure 4 also shows the result of applying modified WDP with fully-smoothed joint probabilities to our new smoothing method *without* significance-based N-gram selection, to test whether the former subsumes the gains from the latter. We see that modified WDP does not render

[4]Thanks to Asela Gunawardana for the use of his REP tool.

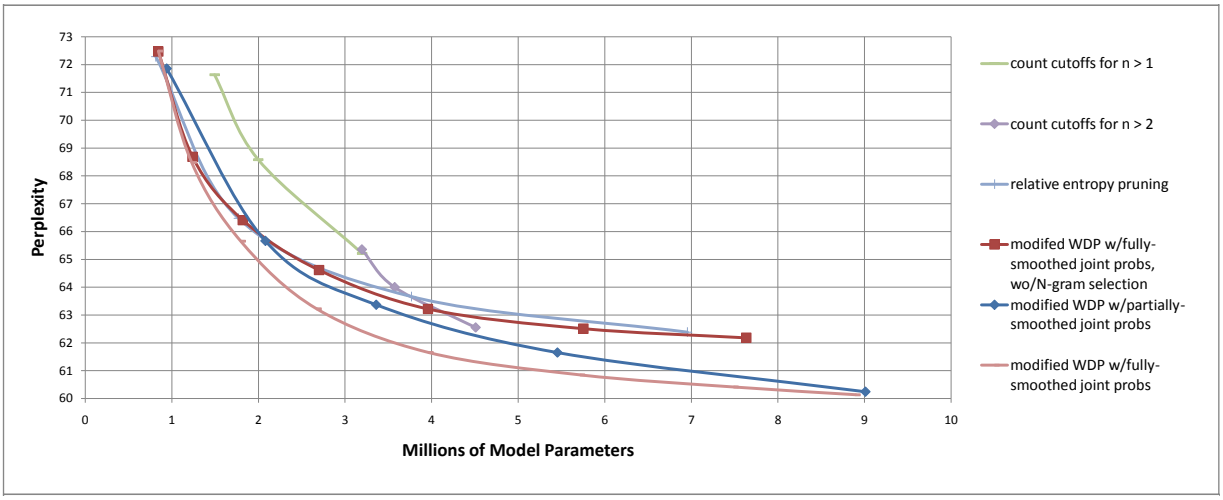


Figure 4: Pruning methods for new smoothing technique with N-gram selection

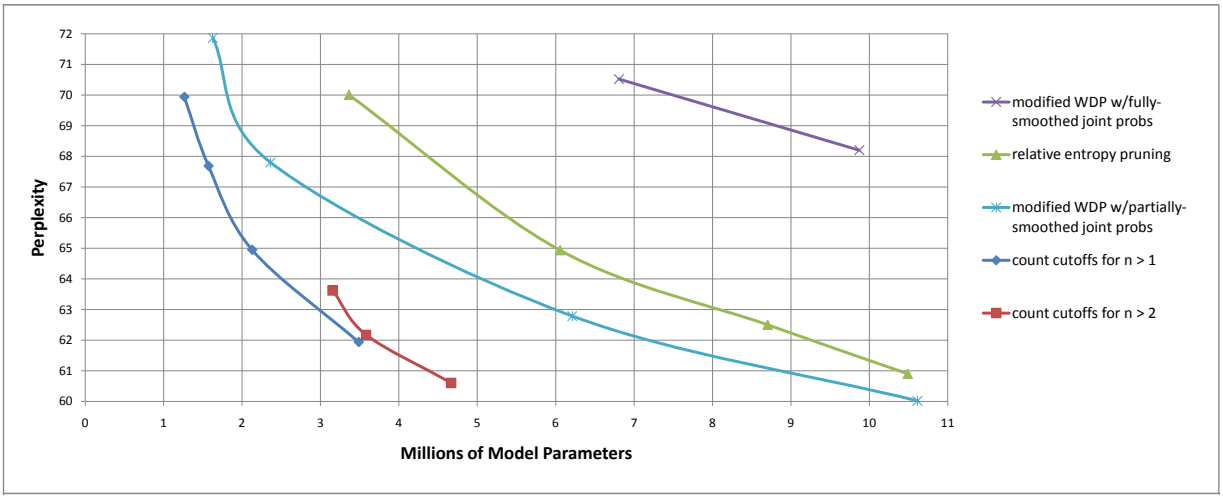


Figure 5: Pruning methods for modified KN smoothing

N-gram selection redundant except at very severe pruning levels, much like REP.

Figure 5 shows the results of applying the same four pruning methods to KN smoothing. Count cutoffs clearly perform the best with KN smoothing. It is interesting to note, however, that—contrary to the results for our new smoothing method—with KN smoothing, modified WDP with partially-smoothed joint probabilities is significantly better than either REP or modified WDP with fully-smoothed joint probabilities. We believe this is due to the fact that the latter two methods both estimate the joint probabilities by chaining the lower-order conditional probabilities from the fully-smoothed model, which in the case of KN smoothing are designed specifically to cover N-grams that have not been observed, and are poor estimates for the probabilities of lower-order N-grams that do occcur in the training data.

Finally, we compared the new smoothing method with N-gram selection and modified WDP with fully-smoothed joint probabilities against modified KN smoothing with count cutoffs, using combinations of pruning parameter values and N-gram order that yielded the best size/perplexity trade-offs. The results are shown in Figure 6. At all model sizes within the range of these experiments, the new method with significance-based N-gram selection and modified WDP had lower perplexity than modifed KN with count cutoffs—up to about 8% lower at greater pruning levels.

This experiment also suggests that the size/perplexity trade-off is easier to optimize for our new combination of smoothing, N-gram selection, and modified WDP, than for KN smoothing with count cut-offs. Table 2 shows the

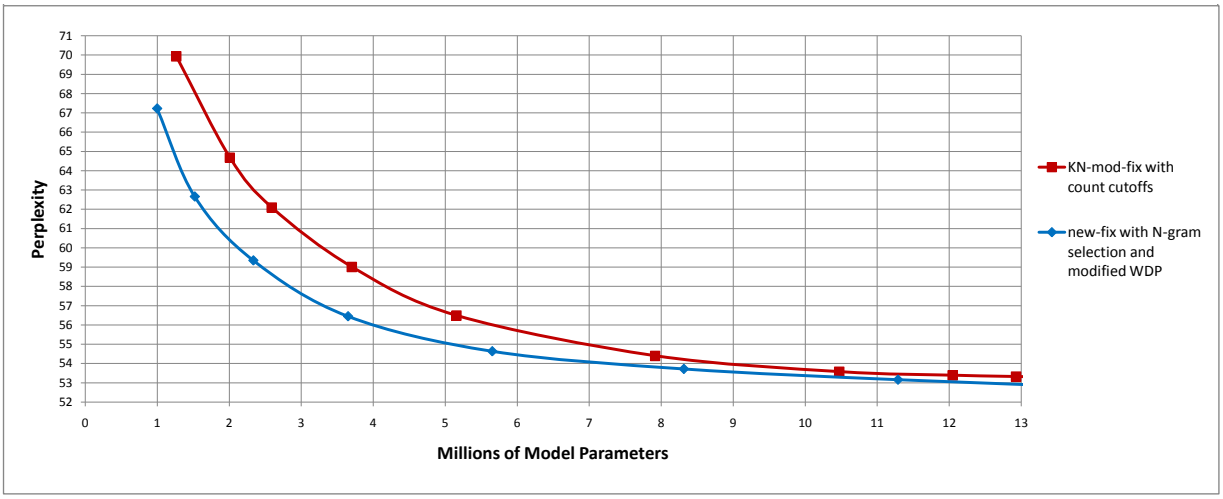


Figure 6: Comparison of two best pruned language models

PP	N	CC	$n >$
69.9	3	4	1
64.7	4	4	1
62.1	4	3	1
59.0	4	2	1
56.5	4	2	2
54.4	4	1	2
53.6	5	1	2
53.4	6	1	2
53.3	7	1	2

Table 2: Optimal pruning parameters for KN-mod-fix with count cutoffs

PP	N	threshold
67.2	10	$10^{-6.5}$
62.7	10	$10^{-6.75}$
59.3	10	$10^{-7.0}$
56.4	10	$10^{-7.25}$
54.6	10	$10^{-7.5}$
53.7	10	$10^{-7.75}$
53.2	10	$10^{-8.0}$

Table 3: Optimal pruning parameters for new-fix with N-gram selection and modified WDP

perplexity (PP), maximum N-gram length (N), count cutoff (CC), and N-gram lengths to which the count cutoffs are applied ($n >$) for the points on the curve for pruned KN in Figure 6. Although some tendencies are discernable, it seems clear that a significant part of the space of combinations of N, CC, and "$n >$" parameter values must be searched to find the best points for trading off perplexity against model size. Table 3 shows maximum N-gram length and pruning threshold values for the points on the corresponding curve for our new approach. Here the situation is much simpler. The best trade-off points are found by varying the pruning threshold, and including in the model all N-grams that pass the pruning threshold, regardless of N-gram length.

6 Conclusions

We have shown that significance-based N-gram selection can simultaneously reduce both model size and perplexity when applied to a number of language model smoothing methods, including the widely-used Katz backoff and absolute discounting methods. We are not aware of any other technique that does this. We also found that, when combined with a new smoothing method and a novel variant of weighted difference pruning, our N-gram selection method outperformed modified Kneser-Ney smoothing—using the best form of pruning we found for that approach—with respect to the trade-off between model size and model quality.

As our next steps, first, we need to verify that the results obtained on a moderate-sized training corpus are repeatable on much larger corpora. Second, we plan to extend this work to incorporate language model size reduction by word clustering, which has been shown by Goodman and Gao (2000) to produce additional gains when combined with previous methods of language model pruning.

References

Brants, Thorsten, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP 2007*, 858–867.

Chen, Stanley F., and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University.

Goodman, Joshua, and Jianfeng Gao. 2000. Language model size reduction by pruning and clustering. In *Proceedings of ICSLP 2000*, 110–113.

Jedynak, Bruno M., and Sanjeev Khudanpur. 2005. Maximum likelihood set for estimating a probability mass function. *Neural Computation* 17, 1–23.

Koehn, Philipp, and Christof Monz. 2006. Manual and automatic evaluation of machine translation between European languages. In *Proceedings of WMT 2006*, 102–121.

Moore, Robert C., and Chris Quirk. 2009. Improved smoothing for N-gram language models based on ordinary counts. In *Proceedings of ACL-IJCNLP 2009*.

Ney, Hermann, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8, 1–38.

Ron, Dana, Yoram Singer, and Naftali Tishby. 1996. The power of amnesia: learning probabilistic automata with variable memory length. *Machine Learning*, 25, 117–149.

Rosenfeld, Ronald, and Xuedong Huang. 1993. Improvements in stochastic language modeling. In *Proceedings of HLT 1993*, 107–111.

Seymore, Kristie, and Ronald Rosenfeld. 1996. Scalable Trigram Backoff Language Models. In *Proceedings of ICSLP 1996*. 232–235.

Stolcke, Andreas. 1998. Entropy-based pruning of backoff language models. In *Proceedings, DARPA News Transcription and Understanding Workshop 1998*, 270–274.

Stream-based Randomised Language Models for SMT

Abby Levenberg
School of Informatics
University of Edinburgh
a.levenberg@sms.ed.ac.uk

Miles Osborne
School of Informatics
University of Edinburgh
miles@inf.ed.ac.uk

Abstract

Randomised techniques allow very big language models to be represented succinctly. However, being batch-based they are unsuitable for modelling an unbounded stream of language whilst maintaining a constant error rate. We present a novel randomised language model which uses an *online perfect hash* function to efficiently deal with unbounded text streams. Translation experiments over a text stream show that our online randomised model matches the performance of batch-based LMs without incurring the computational overhead associated with full retraining. This opens up the possibility of randomised language models which continuously adapt to the massive volumes of texts published on the Web each day.

1 Introduction

Language models (LM) are an integral feature of statistical machine translation (SMT) systems. They assign probabilities to generated hypotheses in the target language informing lexical selection. The most common form of LMs in SMT systems are smoothed n-gram models which predict a word based on a contextual history of $n-1$ words. For some languages (such as English) trillions of words are available for training purposes. This fact, along with the observation that machine translation quality improves as the amount of monolingual training material increases, has lead to the introduction of randomised techniques for representing large LMs in small space (Talbot and Osborne, 2007; Talbot and Brants, 2008).

Randomised LMs (RLMs) solve the problem of representing large, static LMs but they are *batch* oriented and cannot incorporate new data without fully retraining from scratch. This property makes current RLMs ill-suited for modelling the massive volume of textual material published daily on the Web. We present a novel RLM which is capable of incremental (re)training. We use random hash functions coupled with an online perfect hashing algorithm to represent n-grams in small space. This makes it well-suited for dealing with an unbounded stream of training material. To our knowledge this is the first stream-based RLM reported in the machine translation literature. As well as introducing the basic stream-based RLM, we also consider adaptation strategies. Perplexity and translation results show that populating the language model with material chronologically close to test points yields good results. As with previous randomised language models, our experiments focus on machine translation but we also expect that our findings are general and should help inform the design of other stream-based models.

Section 2 introduces the incrementally retrainable randomised LM and section 3 considers related work; Section 4 then considers the question of how unbounded text streams should be modelled. Sections 5 and 6 show stream-based translation results and properties of our novel data-structure. Section 7 concludes the paper.

2 Online Bloomier Filter LM

Our online randomised LM (O-RLM) is based on the dynamic Bloomier filter (Mortensen et al., 2005). It is a variant of the batch-based Bloomier filter LM of Talbot and Brants (2008) which we refer to as the TB-LM henceforth. As with the TB-LM, the O-RLM uses random hash functions to represent n-grams as *fingerprints* which is the main source of space savings for the model.

2.1 Online Perfect Hashing

The key difference in our model as compared to the TB-LM is we use an *online perfect hashing*

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 756–764,
Singapore, 6-7 August 2009.

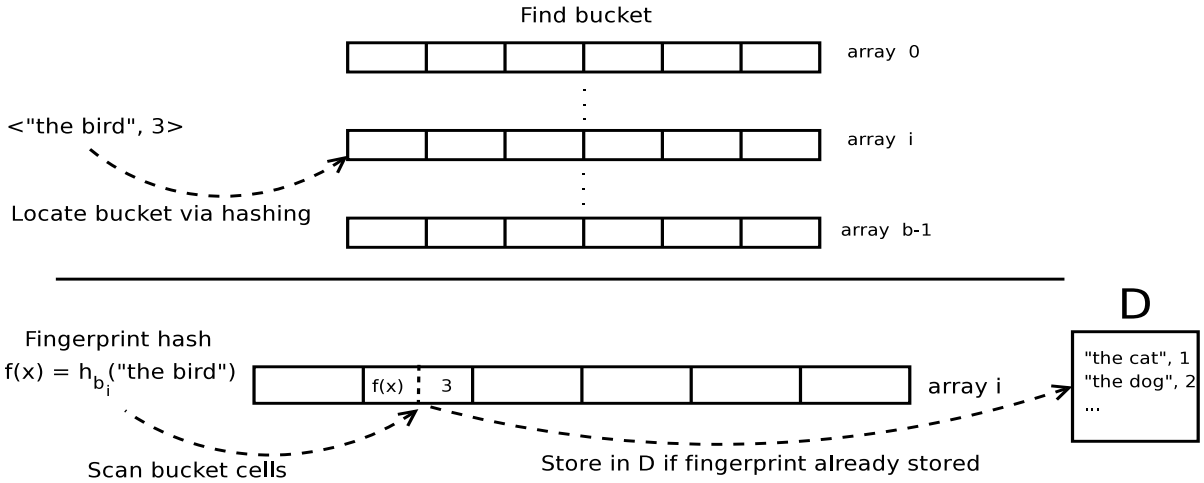


Figure 1: Inserting an n-gram into the dynamic Bloomier filter. Above: an n-gram is hashed to its target bucket. Below: the n-gram is transformed into a fingerprint and the same target bucket is scanned. If a collision occurs that n-gram is diverted to the overflow dictionary; otherwise the fingerprint is stored in the bucket.

function instead of having to precompute the perfect hash offline prior to data insertion.

The online perfect hash function uses two data structures: A and D. A is the main, randomised data structure and is an array of b dictionaries $A_0, \ldots, A_{b-1}$. D is a lossless data structure which handles collisions in A. Each of the dictionaries in A is referred to as a 'bucket'. In our implementation the buckets are equally sized arrays of w-bit cells. These cells hold the fingerprints and values of n-grams (one n-gram-value pair per cell).

To **insert** an n-gram x and associated value $v(x)$ into the model, we select a bucket A_i by hashing x into the range $i \rightarrow [0, \ldots, b-1]$. Each bucket has an associated random hash function, h_{A_i}, drawn from a universal hash function (UHF) family h (Carter and Wegman, 1977), which is then used to generate the n-gram fingerprint: $f(x) = h_{A_i}(x)$.

If the bucket A_i is not full we conduct a scan of its cells. If the fingerprint $f(x)$ is not already encoded in the bucket A_i we add the fingerprint and value to the first empty cell available. We allocate a preset number of the least significant bits of each w-bit cell to hold $v(x)$ and the remaining most significant bits for $f(x)$ but this is arbitrary. Any encoding scheme, such as the packed representation of Talbot and Brants (2008), is viable here.

However, if $f(x) \in A_i$ already (there is a collision) we store the n-gram x and associated value $v(x)$ in the lossless overflow dictionary D instead. D also holds the n-grams that were hashed to any buckets that are already full.

To **query** for the value of an n-gram, we first check if the gram is in the overflow dictionary D. If it is, we return the associated value. Otherwise we query A using the same hash functions and procedure as insertion. If we find a matching fingerprint in the appropriate bucket A_i we have a hit with high probability. **Deletions** and **updates** are symmetric to querying except we reset the cell to the null value or update its value respectively. As with other randomised models we construct queries with the appropriate *sanity checks* to lower the error rate efficiently (Talbot and Brants, 2008).

2.2 Data Insertion

Initially we seed the language model with a large corpus S in the usual manner associated with batch LMs. Then, when processing the stream, we aggregate n-gram counts for some consecutive portion, or *epoch*, of the input stream. We can vary the size of stream window. For example we might batch-up a day or week's worth of material. Intuitively, smaller windows produce results that are sensitive to small variation in the stream, while longer windows (corresponding to data over a longer time period) average out local spikes. The exact window size is a matter of experimentation. In our MT experiments (section 5) we can compute counts within the streaming window exactly but randomised approaches (such as the approximate counting schemes from section 3) can easily be employed instead.

These n-grams and counts are then considered for insertion into the online model. If we decide to insert an n-gram, we either update the count of that n-gram if we previously inserted it or else we insert it as a new entry. Note that there is some probability we may encounter a false positive and update some other n-gram in the model.

2.3 Properties

The online perfect hash succeeds by associating each n-gram with only **one** cell in A rather than having it depend on cells (or bits) which may be shared by other n-grams as with the TB-LM. Since each n-gram's encoding in the model uses distinct bits and is independent of all other events it can not corrupt other n-grams when deleted.

Adding the overflow dictionary D means that we use more space than the TB-LM for the same support. It is shown in Mortensen et al. (2005) that the expected size of D is a small fraction of the total number of events and its space usage comprises less than $O(|S|)$ bits with high probability.

There is a nonzero probability for false positives. Since the overflow dictionary D has no errors, the expected error rate for our dynamic structure is the probability of a random collision in the hash range of each h_{A_i} for each bucket cell compared. In our setup we have

$$\Pr(falsepos) = \frac{|A_i|}{2^{|f(x)|}}$$

where $|f(x)|$ is the number of bits of each w-bit cell used for the fingerprint $f(x)$. w also primarily governs space used in the model. The O-RLM assumes only valid updates and deletions are performed (i.e. we do not remove or update entries that were never inserted prior).

The O-RLM takes time linear to the input size for training and uses worst-case constant time for querying and deletions where the constant is dependent on the number of cells per bucket in A. The number of bucket cells also effects the overall error rate significantly since smaller ranges reduce the probability of a collision. However, too few cells per bucket will result in many full buckets when the bucket hash function is not highly IID.

2.4 Basic RLM Comparisons

Table 1 compares expected versus observed false positive rates for the Bloom filter, TB-LM, and O-RLM obtained by querying a model of approximately 280M events with 100K unseen n-grams.

LM	Expected	Observed	RAM
Lossless	0	0	7450MB
Bloom	0.0039	0.0038	390MB
TB-LM	0.0039	0.0033	640MB
O-RLM	0.0039	0.0031	705MB

Table 1: Example false postive rates and corresponding memory usage for all randomised LMs.

We see the bit-based Bloom filter uses significantly less memory than the cell-based alternatives and the O-RLM consumes more memory than the TB-LM for the same expected error rate.

3 Related Work

3.1 Randomised Language Models

Talbot and Osborne (2007) used a *Bloom filter* (Bloom, 1970) to encode a smoothed LM. A Bloom filter (BF) represents a set S from arbitrary domain U and supports membership queries such as"Is $x \in S$?". The BF uses an array of m bits and k independent UHFs each with range $0, \ldots, m-1$. For insertion, each item is hashed through the k hash functions and the resulting target bits are set to one. During testing, an event $x \in U$ is passed through the same k hash functions and if any bit tested is zero then x was not in the support S.

The *Bloomier filter* directly represents key-value pairs by using a table of cells and a family of k associated hash functions (Chazelle et al., 2004). Each key-value pair is associated with k cells in the table via a perfect hash function. Talbot and Brants (2008) used a Bloomier filter to encode a LM. Before data can be added to the Bloomier filter, a greedy perfect hashing of all entries needs to be computed in advance; this attempts to associate each event in the support with one unique table cell so no other entry collides with it. The procedure can fail and might need to be repeated many times.

Neither of these two randomised language models are suitable for modelling a stream. Given the fact that the stream is of unbounded size, we are forced to delete items if we wish to maintain a constant error rate and account for novel n-grams. However, the Bloom filter LM nor the Bloomier Filter LM support deletions. The bit sharing of the Bloom filter (BF) LM (Talbot and Osborne, 2007) means deletions may corrupt shared stored events. The Bloomier filter LM (Talbot and Brants, 2008) has a precomputed matching of keys shared between a constant number of cells in the filter array.

Deleting items from a Bloomier Filter without recomputing the perfect hash will corrupt it.

3.2 Probabilistic Counting

Concurrent work has used approximate counting schemes based on Morris (1978) to estimate in small space frequencies over a high volume input text stream (Van Durme and Lall, 2009; Goyal et al., 2009). The space savings are due to compact storage of counts and retention of only a small subset of the available n-grams in the data stream. Since the final LMs are still lossless (modulo counts), the resulting LM needs significant space. It is trivial to use probabilistic counting within our framework.

3.3 Compact Exact Language Models

Randomised algorithms are not the only compact representation schemes. Church et al. (2007) looked at Golomb Coding and Brants et al. (2007) used tries in a distributed setting. These methods are less succinct than randomised approaches.

3.4 Adaptive Language Models

There is a large literature on adaptive LMs from the speech processing domain (Bellegarda, 2004). The primary difference between the O-RLM and other adaptive LMs is that we add and remove n-grams from the model instead of adapting only the parameters of the current support set.

3.5 Domain adaptation in Machine Translation

Within MT there has been a variety of approaches dealing with domain adaption (for example (Wu et al., 2008; Koehn and Schroeder, 2007). Typically LMs are interpolated with one another, yielding good results. These models are usually statically trained, exact and unable to deal with an unbounded stream of monolingual data. Domain adaptation has similarities with streaming, in that our stream may be non-stationary. A crucial difference however is that the stream is of unbounded length, whereas domain adaptation usually assumes some finite and fixed training set.

4 Stream-based translation

Streaming algorithms have numerous applications in mainstream computer science (Muthukrishnan, 2003) but to date there has been very little awareness of this field within computational linguistics.

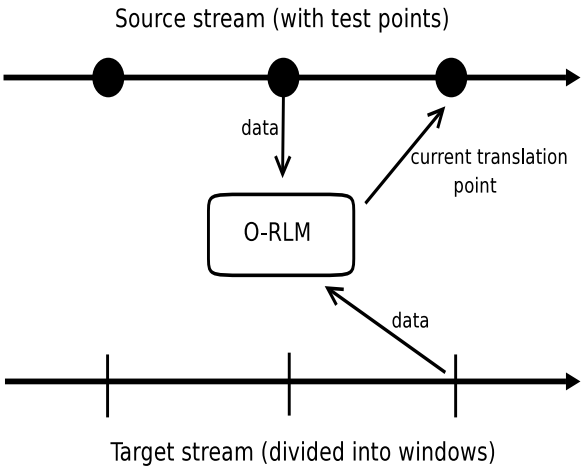


Figure 2: Stream-based translation. The online RLM uses data from the target stream and the last test point in the source stream for adaptation.

A *text stream* can be thought of as a unbounded sequence of documents that are time-stamped and we have access to them in strict chronological order. The volume of the stream is so large we can afford only a limited number of passes over the data (typically one).

Text streams naturally arise on the Web when millions of new documents are published each day in many languages. For instance, 18 thousand websites continuously publish news stories in 40 languages and there are millions of multilingual blog postings per day. There are over 30 billion e-mails sent daily and social networking sites, including services such as Twitter, generate an adundance of textual data in real time. Web crawlers that spidered all these new documents would produce an unbounded input stream.

The stream-based translation scenario is as follows: we assume that each day we see a source stream of many new newswire stories that need translation. We also assume a stream of newswire stories in the target language. Intuitively, since the concurrent streams are from the same domain, we can use the contexts provided in the target stream to help with the translation of the source stream (Figure 2). From a theoretical perspective, since we cannot represent the entirety of the stream and wish to maintain a constant error rate, we are forced to throw some information away.

Given that the incoming text stream contains far too much data to store in its entirety an immediate question we would like to answer is: within our LM, which subset of the target text stream should

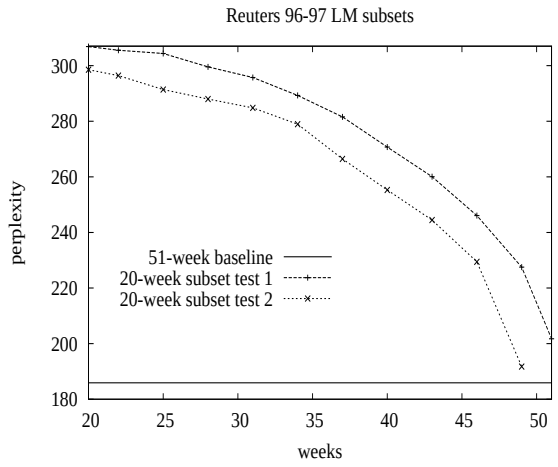


Figure 3: Perplexity results using streamed data. Perplexity decreases as we retrain LMs using data chronologically closer to the (two) test dates.

we represent in our model?

Using perplexity, we investigated this question using a text stream based on Reuter's RCV1 text collection (Rose et al., 2002). This contains 800k time-stamped newswire stories from a full calender year (8.20.1996 - 8.19.1997). We used the SRILM (Stolcke, 2002) to construct an exact trigram model built using all the RCV1 data with the exception of the final week which we held out as test data. This served as an oracle since we store all of the stream.

We then trained multiple exact LMs of much smaller sizes, coined *subset LMs*, to simulate memory constraints. For a given date in the RCV1 stream, these subset LMs were trained using a fixed window of previously seen documents up to that data. Then we obtained perplexity results for each subset LM against our test set.

Figure 3 shows an example. For this experiment subset LMs were trained using a sliding window of 20 weeks with the window advancing over a period of three weeks each time. The two arcs correspond to two different test sets drawn from different days. The arcs show that **recency** has a clear effect: populating LMs using material closer to the test data date produces improved perplexity performance. The LM chronologically closest to a given test set has perplexity closest to the results of the significantly larger baseline LM which uses all the stream. As expected, using all of the data yields the lowest perplexity.

We note that this is a robust finding, since we also observe it in other domains. For example, we conducted the same tests over a stream of 18 billion tokens drawn from 80 million time-stamped blog posts downloaded from the web with matching results. The effect of recency on perplexity has also been observed elsewhere (see, for example, Rosenfeld (1995) and Whittaker (2001)).

Epoch	Stream Window
1	08.20.1996 to 01.01.1997
2	01.02.1997 to 04.23.1997
3	04.24.1997 to 08.18.1997

Table 2: The stream timeline is divided into windowed epochs for our recency experiments.

Our experiments show that a possible way to tackle stream-based translation is to always focus the attention of the LM on the most recent part of the stream. This means we remove data from the model that came from the receding parts of the stream and replace it with the present.

5 SMT Experiments

5.1 Experimental Setup

We used publicly available resources for all our tests: for decoding we used Moses (Koehn and Hoang, 2007) and our parallel data was taken from the Spanish-English section of Europarl. For test material, we translated 63 documents (800 sentences) from three randomly selected dates spaced throughout the RCV1 year (January 2nd, April 24, and August 19).[1] This effectively divided the stream into three *epochs* between the test dates (table 2). We held out 300 sentences for minimum error rate training (MERT) (Och, 2003) and optimised the parameters of the feature functions of the decoder for each experimental run.

The RCV1 is not a large corpus when compared to the entire web but it is multilingual, chronological, and large enough to enable us to test the effect of recency in a translation setting.

5.2 Adaption

We looked at a number of ways of adapting the O-RLM:

1. (**Random**) Randomly sample the stream and for each new n-gram encountered, insert

[1] As RCV1 is not a parallel corpus we translated the reference documents ourselves. This parallel corpus is available from the authors.

Order	Full	Epoch 1	Epoch 3
1	1.25M	0.6M	0.7M
2	14.6 M	6.8M	7.0M
3	50.6 M	21.3M	21.7M
4	90.3 M	34.8M	35.4M
5	114.7M	41.8M	42.6M
Total	271.5M	105M	107.5M

Table 3: Distinct n-grams (in millions) encountered in the full stream and example epochs.

it and remove some previously inserted n-gram, irrespective of whether it was ever requested by the decoder or is a prefix.

2. (**Conservative**) For each new n-gram encountered in the stream, insert it in the filter and remove one previously inserted n-gram which was never requested by the decoder. To preserve consistency we do not remove lower-order grams that are needed to estimate backoff probability for higher-order smoothing. Counts are updated for n-grams already in the model if the new count observed is larger than the current one.

3. (**Severe**) Differs from the conservative approach only in that we delete *all* unused n-grams (i.e. all those not requested by the decoder in the previous translation task) from the O-RLM before adapting with data from the stream. This means the data structure is sparsely populated for all runs.

All the TB-LMs and O-RLMs were unpruned 5-gram models and used *Stupid-backoff* smoothing (Brants et al., 2007) [2] with the backoff parameter set to 0.4 as suggested. The number of distinct n-grams encountered in the stream for two epochs is shown in Table 3.

Table 6 shows translation results using these adaption strategies. In practice, the random approach does not work while the conservative and severe adaption techniques produce equivalent results due to the small proportion of data in the model that is queried during decoding. All the MT experiments that follow use the severe method and the overflow dictionary always holds less than 1% of the total elements in the model.

[2] Smoothing text input data streams poses an interesting problem we hope to investigate in the future.

Date	Lossless	TB-LM	O-RLM
Jan	37.83	37.12	37.17
Apr	34.88	34.21	34.79
Aug	29.05	28.52	28.44
Avg	33.92	33.28	33.46

Table 4: Baseline translation results in BLEU using data from the first stream epoch with a lossless LM (4.5GB RAM), the TB-LM and the O-RLM (300MB RAM). All LMs are static.

5.3 Training Regimes

We now consider stream-based translation. Our first *naive approach* is to continually add new data from the stream to the training set without deleting anything. Given a constant memory bound this strategy only increases the error rate over time as discussed. Our second, computationally demanding approach is, before each test point, to rebuild the TB-LM from scratch using the stream data from the most recent epoch as the training set. This is *batch retraining*. The final approach incrementally retrains online. This utilizes the same training data as above (the stream data from the last epoch) but instead of full retraining it replaces n-grams currently in the model with unseen n-grams and counts encountered in the data stream.

5.4 Streaming Translation Results

Each table shows translation results for the three different test times in the stream. All results reported use the case-sensitive BLEU score.

For our baselines we use *static* LMs trained on the first epoch's data to test all three translation points in the source stream. This is the traditional approach. We trained an exact, modified Kneser-Ney smoothed LM (here we do not enforce a memory constraint) and also used the TB-LM and O-RLM to verify our structures adequecy. Results are shown in table 4. The exact model gives better performance overall due to the more sophisticated smoothing used.

Table 5 shows results for a set of *stream-based* LMs using the TB-LM and the O-RLM with memory bounds of 200MB and 300MB. As expected, the naive models performance degrades over time as we funnel more data into the TB-LM and the error rises. The batch retrained TB-LMs and O-RLMs have constant error rates of $\frac{1}{2^8}$ and $\frac{1}{2^{12}}$ and so outperform the naive approach. Since the training data is identical we see (approximately) equal

	Naive TB-LM		Batch Retrained TB-LM		O-RLM	
Date	200MB	300MB	200MB	300MB	200MB	300MB
Jan	35.94	37.12	35.94	37.12	36.44	37.17
Apr	33.55	35.79	36.01	35.99	35.87	36.10
Aug	22.44	26.07	28.97	29.38	29.00	29.18
Avg	30.64	32.99	33.64	34.16	33.77	34.15

Table 5: Translation results for stream-based LMs in BLEU. Performance degrades with time using the Naive approach. The batch retrained TB-LM and stream-based O-RLM use constant error rates of $\frac{1}{2^8}$ and $\frac{1}{2^{12}}$.

performance from the batch retrained and online models. We also see some improvement compared to the static baselines when the LMs use the most recent data from the target language stream with respect to the current translation point.

The key difference is that each time we batch retrain the TB-LM, we must compute a perfect hashing of the new training set. This is computationally demanding since the perfect hashing algorithm uses Monte Carlo randomisation which fails routinely and must be repeated. To make the algorithm tractable the training data set must be divided into lexically sorted subsets as well. This requires extra passes over the data which may not be trivial in a streaming environment.

In contrast, the O-RLM is incrementally retrained online. This makes it more resource efficient since we find bits in the model for the n-grams dynamically without using more memory than we intially set. Note that even though the O-RLM is theoretically less space efficient than the TB-LM, when using the same amount of memory translation performance is comparable.

6 O-RLM Properties

The previous experiments confirm that the O-RLM can be employed as a LM in an SMT setting but it is useful to get insight into the intrinsic properties of the data structure. Many of the properties of the model, such as the number of bits per fingerprint, follow directly from the TB-LM but the relationship between the overflow dictionary and the randomised buckets is novel.

Figures 4 and 5 shows properties of the O-RLM while varying only the number of cells in each bucket and keeping all other model parameters constant. We test membership of n-grams in an unseen corpus against those stored in the table. Our tests were conducted over a larger stream of 1.25B n-grams from the Gigaword corpus(Graff, 2003). We set our space usage to match the 3.08 bytes per n-gram reported in Talbot and Brants (2008) and held out just over 1M unseen n-grams to test the error rates of our models.

Date	Severe	Random	Conservative
Jan	36.44	36.44	36.44
Apr	35.87	31.08	35.51
Aug	29.00	19.31	29.14
Avg	33.77	29.11	33.70

Table 6: Adaptation results measured in BLEU. Random deletions degrade performance when adapting a 200MB O-RLM.

In Figure 4 we see a direct correlation between model error and cells per buckets. As the number of cells decreases the false positive rate drops as well since fewer cells to compare against per bucket means a lower chance of producing collisions. If the range is decreased too much though more data is diverted to the overflow dictionary due to many buckets reaching capacity when inserting and adapting. Clearly this is less space efficient. Figure 5 shows the relationship between the percent of data in the overflow dictionary and the total cells per bucket.

7 Conclusions

Our experiments have shown that for stream-based translation, using recent data can benefit performance but simply adding entries to a randomised representation will only reduce translation performance over time. We have presented a novel randomised language model based on dynamic perfect hashing that supports online insertions and deletions. As a consequence, it is considerably faster and more efficient than batch retraining.

While not advocating the idea that only small amounts of data are needed for language mod-

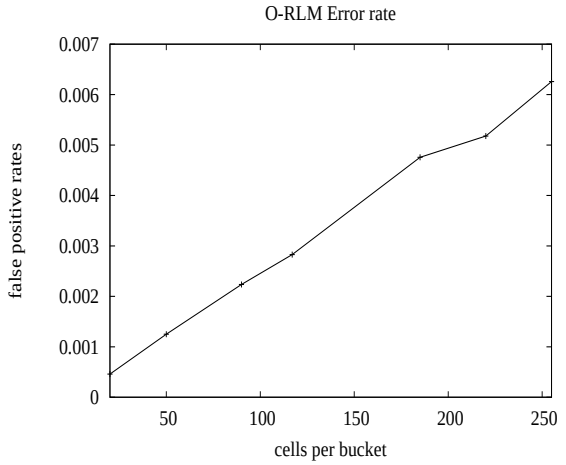


Figure 4: The O-RLM error rises in correlation with the number of cells per bucket.

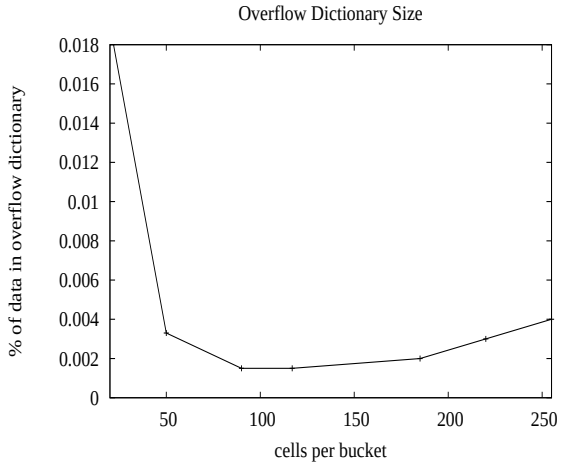


Figure 5: Too few cells per bucket causes a higher percentage of the data to be stored in the overflow dictionary due to full buckets.

elling, within a bounded amount of space our results show that it is better to have a low error rate and store a wisely chosen fraction of the data than having a high error rate and storing more of it. Clearly tradeoffs will vary between applications.

This is the first stream-based randomised language model and associated machine translation system reported in the literature. Clearly there are many interesting open questions for future work. For example, can we use small randomised representations called *sketches* to compactly represent side-information on the stream telling us which aspects of it we should insert into our data? How can we efficiently deal with smoothing in this setting? Our adaptation scheme is simple and our data stream is tractable. Currently we are conducting tests over much larger, higher variance text streams from crawled blog data. In the future we will also consider randomised representations of other adaptive LMs in the literature using a static background LM in conjunction with our online one. We ultimately hope to deploy large-scale LMs which continuously adapt to the vast amount of material published on the Web without incurring significant computational overhead.

Acknowledgements

The authors would like to thank David Talbot, Adam Lopez and Phil Blunsom for their valuable comments and insight. This work was supported in part under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022.

References

Jerome R. Bellegarda. 2004. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42:93–108.

Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867.

J. Lawrence Carter and Mark N. Wegman. 1977. Universal classes of hash functions (extended abstract). In *STOC '77: Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 106–112, New York, NY, USA. ACM Press.

Bernard Chazelle, Joe Kilian, Ronitt Rubinfeld, and Ayellet Tal. 2004. The bloomier filter: an efficient data structure for static support lookup tables. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 30–39, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Kenneth Church, Ted Hart, and Jianfeng Gao. 2007. Compressing trigram language models with Golomb coding. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 199–207, Prague, Czech Republic, June. Association for Computational Linguistics.

Amit Goyal, Hal Daumé III, and Suresh Venkatasubramanian. 2009. Streaming for large scale NLP: Language modeling. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, Boulder, CO.

David Graff. 2003. English Gigaword. Linguistic Data Consortium (LDC-2003T05).

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic, June. Association for Computational Linguistics.

Robert Morris. 1978. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842.

Christian Worm Mortensen, Rasmus Pagh, and Mihai Pătraçcu. 2005. On dynamic range reporting in one dimension. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 104–111, New York, NY, USA. ACM.

S. Muthukrishnan. 2003. Data streams: algorithms and applications. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 413–413, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Morristown, NJ, USA. Association for Computational Linguistics.

Tony Rose, Mark Stevenson, and Miles Whitehead. 2002. The reuters corpus volume 1 - from yesterdays news to tomorrows language resources. In *In Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 29–31.

Ronald Rosenfeld. 1995. Optimizing lexical and n-gram coverage via judicious use of linguistic data. In *In Proc. European Conf. on Speech Technology*, pages 1763–1766.

A. Stolcke. 2002. Srilm – an extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing, 2002*.

David Talbot and Thorsten Brants. 2008. Randomized language models via perfect hash functions. In *Proceedings of ACL-08: HLT*, pages 505–513, Columbus, Ohio, June. Association for Computational Linguistics.

David Talbot and Miles Osborne. 2007. Smoothed Bloom filter language models: Tera-scale LMs on the cheap. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 468–476.

Benjamin Van Durme and Ashwin Lall. 2009. Probabilistic counting with randomized storage. In *Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, Pasadena, CA, July.

E. W. D. Whittaker. 2001. Temporal adaptation of language models. In *In Adaptation Methods for Speech Recognition, ISCA Tutorial and Research Workshop (ITRW)*, pages 203–206.

Hua Wu, Haifeng Wang, and Chengqing Zong. 2008. Domain adaptation for statistical machine translation with domain dictionary and monolingual corpora. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 993–1000. Coling 2008 Organizing Committee, August.

Integrating sentence- and word-level error identification for disfluency correction

Erin Fitzgerald
Johns Hopkins University
Baltimore, MD, USA
erinf@jhu.edu

Frederick Jelinek
Johns Hopkins University
Baltimore, MD, USA
jelinek@jhu.edu

Keith Hall
Google Inc.
Zürich, Switzerland
kbhall@google.com

Abstract

While speaking spontaneously, speakers often make errors such as self-correction or false starts which interfere with the successful application of natural language processing techniques like summarization and machine translation to this data. There is active work on reconstructing this errorful data into a clean and fluent transcript by identifying and removing these simple errors.

Previous research has approximated the potential benefit of conducting word-level reconstruction of simple errors only on those sentences known to have errors. In this work, we explore new approaches for automatically identifying speaker construction errors on the utterance level, and quantify the impact that this initial step has on word- and sentence-level reconstruction accuracy.

1 Introduction

A system would accomplish reconstruction of its spontaneous speech input if its output were to represent, in flawless, fluent, and content-preserving text, the message that the speaker intended to convey. While full speech reconstruction would likely require a range of string transformations and potentially deep syntactic and semantic analysis of the errorful text (Fitzgerald, 2009), in this work we will attempt only to resolve less complex errors, correctable by deletion alone, in a given manually-transcribed utterance.

The benefit of conducting word-level reconstruction of simple errors only on those sentences known to have errors was approximated in (Fitzgerald et al., 2009). In the current work, we explore approaches for automatically identifying speaker-generated errors on the utterance level, and calculate the gain in accuracy that this initial step has on word- and sentence-level accuracy.

1.1 Error classes in spontaneous speech

Common simple disfluencies in sentence-like utterances (SUs) include *filler words* (i.e., "um", "ah", and discourse markers like "you know"), as well as speaker edits consisting of a *reparandum*, an *interruption point (IP)*, an optional *interregnum* (like "I mean"), and a *repair* region (Shriberg, 1994), as seen in Figure 1.

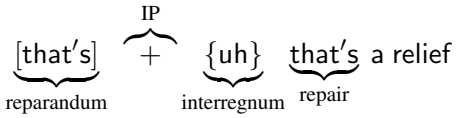


Figure 1: Typical edit region structure.

These reparanda, or *edit regions*, can be classified into three main groups:

1. In a **repetition** (above), the repair phrase is approximately identical to the reparandum.
2. In a **revision**, the repair phrase alters reparandum words to correct the previously stated thought.

   EX1: but [when he] + {i mean} when she put it that way

   EX2: it helps people [that are going to quit] + that would be quitting anyway
3. In a **restart fragment** an utterance is aborted and then restarted with a new train of thought.

   EX3: and [i think he's] + he tells me he's glad he has one of those

   EX4: [amazon was incorporated by] {uh} well i only knew two people there

In *simple cleanup* (a precursor to full speech reconstruction), all detected filler words are deleted, and the reparanda and interregna are deleted while the repair region is left intact. This is a strong initial step for speech reconstruction, though more

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 765–774,
Singapore, 6-7 August 2009.

```
1 he  that 's  uh that 's a relief
2  E    E   E  FL   -   - -    -
3 NC   RC  RC  FL   -   - -    -
```

Figure 2: Example of word class and refined word class labels, where – denotes a non-error, FL denotes a filler, E generally denotes reparanda, and RC and NC indicate rough copy and non-copy speaker errors, respectively. Line 3 refines the labels of Line 2.

complex and less deterministic changes may be required for generating fluent and grammatical speech text in all cases.

1.2 Related Work

Stochastic approaches for simple disfluency detection use features such as lexical form, acoustic cues, and rule-based knowledge. State-of-the-art methods for edit region detection such as (Johnson and Charniak, 2004; Zhang and Weng, 2005; Kahn et al., 2005; Honal and Schultz, 2005) model speech disfluencies as a noisy channel model, though direct classification models have also shown promise (Fitzgerald et al., 2009; Liu et al., 2004). The final output is a word-level tagging of the error condition of each word in the sequence, as seen in line 2 of Figure 2.

The Johnson and Charniak (2004) approach, referred to in this document as JC04, combines the noisy channel paradigm with a tree-adjoining grammar (TAG) to capture approximately repeated elements. The TAG approach models the crossed word dependencies observed when the reparandum incorporates the same or very similar words in roughly the same word order, which JC04 refer to as a *rough copy*. Line 3 of Figure 2 refines "edits" (E) into rough copies (RC) and *non-copies* (NC).

As expected given the assumptions of the TAG approach, JC04 identifies repetitions and most revisions in spontaneous data, but is less successful in labeling false starts and other speaker self-interruptions without cross-serial correlations. These non-copy errors hurt the edit detection recall and overall accuracy.

Fitzgerald et al. (2009) (referred here as FHJ) used conditional random fields (CRFs) and the Spontaneous Speech Reconstruction (SSR) corpus (Fitzgerald and Jelinek, 2008) corpus for word-level error identification, especially targeting improvement of these non-copy errors. The CRF was trained using features based on lexical, language model, and syntactic observations along with features based on JC04 system output.

Alternate experimental setup showed that training and testing only on SUs known from the labeled corpus to contain word-level errors yielded a notable improvement in accuracy, indicating that the described system was falsely identifying many non-error words as errors.

Improved sentence-level identification of errorful utterances was shown to help improve word-level error identification and overall reconstruction accuracy. This paper describes attempts to extend these efforts.

2 Approach

2.1 Data

We conducted our experiments on the recently released Spontaneous Speech Reconstruction (SSR) corpus (Fitzgerald and Jelinek, 2008), a medium-sized set of disfluency annotations atop Fisher conversational telephone speech data (Cieri et al., 2004)[1]. Advantages of the SSR data include

- aligned parallel original and cleaned sentences
- several levels of error annotations, allowing for a coarse-to-fine reconstruction approach
- multiple annotations per sentence reflecting the occasional ambiguity of corrections

As reconstructions are sometimes non-deterministic, the SSR provides two manual reconstructions for each utterance in the data. We use these dual annotations to learn complementary approaches in training and to allow for more accurate evaluation.

The Spontaneous Speech Reconstruction corpus is partitioned into three subcorpora: 17,162 training sentences (119,693 words), 2,191 sentences (14,861 words) in the development set, and 2,288 sentences (15,382 words) in the test set. Approximately 17% of the total utterances contain a reparandum-type error. In constructing the data, two approaches were combined to filter out the utterances considered most likely to be errorful (6,384 in total) and only those SUs were manually reconstructed. However the entire data set was included in the distribution – and used in training for this work – to maintain data balance.

[1]The Spontaneous Speech Reconstruction corpus can be downloaded from http://www.clsp.jhu.edu/PIRE/ssr.

The training of the TAG model for JC04, used as a feature in this work, requires a very specific data format, and thus is trained not with SSR but with Switchboard (SWBD) data (Godfrey et al., 1992). Key differences in these corpora, besides the granularity and form of their annotations, include:

- SSR aims to correct speech output, while SWBD edit annotation aims to identify reparandum structures specifically. SSR only marks those reparanda which annotators believe must be deleted to generate a grammatical and content-preserving reconstruction.
- SSR includes more complex error identification and correction, not considered in this work.

While the SWBD corpus has been used in some previous simple disfluency labeling work (e.g., Johnson and Charniak, 2004; Kahn et al., 2005), we consider the SSR for its fine-grained error annotations.

3 Identifying poor constructions

Prior to reconstruction, it is to our advantage to automatically identify poorly constructed sentences, defined as being ungrammatical, incomplete, or missing necessary sentence boundaries. Accurately extracting ill-formed sentences prior to subsentential error correction helps to minimize the risk of information loss posed by unnecessarily and incorrectly reconstructing well-formed text.

To evaluate the efforts described below, we manually label each SU s in the SSR test set S (including those not originally annotated with reconstructions but still included in the SSR distribution) as *well-formed* or *poorly-formed*, forming the set of poorly constructed SUs $P \subset S$, $|P| = 531$ and $|S| = 2288$ utterances.

To identify speaker errors on the sentence level, we consider and combine a collection of features into a single framework using a maximum entropy model (implemented with the Daumé III (2004) MEGA Model toolkit).

3.1 SU-level error features

Six feature types are presented in this section.

- Features #1 and #2 are the two methods included in a similar though less exhaustive effort by (Fitzgerald and Jelinek, 2008) in error filtering for the creation of the SSR corpus itself.
- Feature types #3 and #4 extract features from automatic parses assigned to the given sentence. It is expected that these parses will contain some errors and the usefulness of these features may be parser-specific. The value of these features though is the consistent, if not always accurate, treatment of similar construction errores given a particular state-of-the-art parser.
- Feature type #5 investigates the relationship between the probability of a SU-internal error and the number of words it contains.
- Feature type #6 serves to bias the probability against assigning a backchannel acknowledgement SU as an error instance.

Feature #1 (JC04): Consider only sentences with JC04 detected edit regions. This approach takes advantage of the high precision, low recall JC04 disfluency detection approach described in Section 1.2. We apply the out-of-box JC04 system and consider any sentence with one or more labeled reparanda as a "poor" indicator. Since speakers repairing their speech once are often under a higher cognitive load and thus more likely to make more serious speech errors (in other words, there is a higher probability of making an error given that an error has already been made (Bard et al., 2001)). This is a reasonable first order approach for finding deeper problems.

Feature #2 (HPSG): Use deep linguistic parsers to confirm well-formedness. Statistical context-free parsers are highly robust and, due to smoothing, can assign a non-zero probability syntactic structure even for text and part-of-speech sequences never seen during training. However, sometimes no output is preferable to highly errorful output. Hand-built rule-based parsers can produce extremely accurate and context-sensitive syntactic structures, but are also brittle and do not adapt well to never before seen input. We use this inflexibility to our advantage.

Head-driven Phrase Structure Grammar (HPSG) is a deep-syntax phrase structure grammar which produces rich, non-context-free syntactic analyses of input sentences based on a collection of carefully constructed rules and lexical item structures (Pollard and Sag, 1994; Wahlster, 2000). Each utterance is parsed using

the PET deep parser produced by the inter-institutional DELPH-IN group[2]. The manually compiled English Resource Grammar (ERG) (Flickinger, 2002) rules have previously been extended for the Verbmobil (Wahlster, 2000) project to allow for the parsing of basic conversational elements such as SUs with no verb or basic backchannel acknowledgements like "last thursday" or "sure", but still produce strict HPSG parses based on these rules. We use the binary result of whether or not each SU is parsable by the HPSG ERG as binary indicator functions in our models.

There has been some work on producing partial parses for utterances for which a full HPSG analysis is not deemed possible by the grammar (Zhang et al., 2007). This work has shown early promise for identifying coherent substrings within errorful SUs given subjective analysis; as this technology progresses, HPSG may offer informative subsentential features for word-level error analysis as well.

Feature #3 (Rules): Mark unseen phrase rule expansions. Phrase-based parses are composed of a recursive sequence of non-terminal (NT) rule expansions, such as those detailed for the example parse shown in Figure 3. These rules are learned from training data such as the Switchboard treebank, where telephone conversation transcripts were manually parsed. In many statistical parsers, new structures are generated based on the relative frequencies of such rules in the training treebank, conditioned on the terminal words and some local context, and the most probable parse (roughly the joint probability of its rule expansions) is selected.

Because parsers are often required to produce output for words and contexts never seen in the training corpus, smoothing is required. The Charniak (1999) parser accomplishes this in part through a *Markov grammar* which works top-down, expanding rules to the left and right of an expansion head M of a given rule. The non-terminal (NT) M is first predicted from the parent P, then – in order – L_1 through L_m (stopping symbol '#') and R_1 through R_n (again '#'), as shown in Equation 1.

$$\text{parent } P \rightarrow \#L_m \dots L_1 M R_1 \dots R_n \# \qquad (1)$$

In this manner, it is possible to produce rules never before seen in the training treebank. While this may be required for parsing grammatical sentences with rare elements, this SU-level error prediction feature indicates whether the automatic parse for a given SU includes an expansion never seen in the training treebank. If an expansion rule in the one-best parse was not seen in training (here meaning in the SWBD treebank after EDITED nodes have been removed), the implication is that new rule generation is an indicator of a speaker error within a SU.

Feature #4 (C-comm): Mark unseen rule c-commanding NTs. In X' theory (Chomsky, 1970), lexical categories such as nouns and verbs are often modified by a specifier (such as the DT "a" modifying the NN "lot" in the NP_3 phrase in Figure 3 or an auxiliary verb for a verb in a verb phrase (VBZ for VP_3) and a complement (such as the object of a verb NP_3 for VBG in the phrase VP_3).

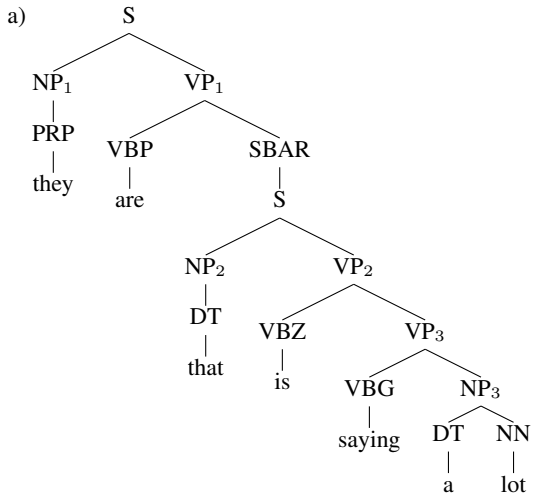
In each of these cases, an NT tree node A has the following relationship with a second NT P:

- Neither does node A *dominate* P nor node P dominate A, (i.e., neither is directly above the other in the parse tree), and
- Node A immediately *precedes* P in the tree (precedence is represented graphically in left-to-right order in the tree).

Given these relationships, we say that A *locally c-commands* P and its descendants. We further extend this definition to say that, if node $\hat{A}$ is the only child of node A (a unary expansion) and A locally c-commands P, then $\hat{A}$ locally c-commands P (so both [SBAR $\rightarrow$ S] and [S $\rightarrow$ NP_2 VP_2] are c-commanded by VBP). See Figure 3 for other examples of non-terminal nodes in c-commanding relationships, and the phrase expansion rule they c-command.

The c-command relationship is fundamental in syntactic theory, and has uses such as predicting the scope of pronoun antecedents. In this case, however, we use it to describe two nodes which are in a specifier–category relationship or a category–complement relationship (e.g., subject–verb and verb–object, respectively). This is valuable to us because it takes advantage of a weakness of statistical parsers: the context used to condition the probability of a given rule expansion generally does not reach beyond dominance relationships, and thus parsers rarely penalize for the juxtaposition of A c-commanding P and its children as

[2]The DEep Linguistic Processing with HPSG INitiative (see http://www.delph-in.net/)



b) Rules expansions:
S → NP VP
NP_1 → PRP
VP_1 → VBP SBAR
SBAR → S
S → NP_2 VP_2
NP_2 → DT
VP_2 → VBZ VP
VP_3 → VBG NP
NP_3 → DT NN

c) Rule expansions + c-commanding NT:	
S → NP VP	*no local c-command*
NP_1 → PRP	*no local c-command*
VP_1 → V SBAR	NP_1
SBAR → S	VBP
S → NP_2 VP_2	VBP
NP_2 → DT	*no local c-command*
VP_2 → VBZ VP	NP_2
VP_3 → VBG NP	VBZ
NP_3 → DT NN	VBG

Figure 3: The automatically generated parse (a) for an errorful sentence-like unit (SU), with accompanying rule expansions (b) and local c-commands (c). Non-terminal indices such as NP_2 are for reader clarification only and are not considered in the feature extraction process.

long as they have previously seen NT type A preceding NT type P. Thus, we can use the children of a parent node P as a way to enrich a NT type P and make it more informative.

For example, in Figure 3, the rule [S → NP_2 VP_2] is routinely seen in the manual parses of the SWBD treebank, as is [VP_1 → VBP SBAR]. However, it is highly unusual for VBP to immediately precede SBAR or S when this rule expands to NP_2 VP_2. So, not only does SBAR/S complement VBP, but a very specific type of [SBAR/S → NP VP] is the complement of VBP. This conditional infrequency serves as an indication of deeper structural errors.

Given these category relationship observations, we include in our maximum entropy model a feature indicating whether a given parse includes a c-command relationship not seen in training data.

Feature #5 (Length): Threshold sentences based on length. Empirical observation indicates that long sentences are more likely to contain speaker errors, while very short sentences tend to be backchannel acknowledgments like "yeah" or "I know" which are not considered errorful. Oviatt (1995) quantifies this, determining that the disfluency rate in human-computer dialog increases roughly linearly with the number of words in an utterance.

The length-based feature value for each sentence therefore is defined to be the number of word tokens in that sentence.

Feature #6 (Backchannel): Bias backchannel acknowledgements as non-errors A backchannel acknowledgement is a short sentence-like unit (SU) which is produced to indicate that the speaker is still paying attention to the other speaker, without requesting attention or adding new content to the dialog. These SUs include "uh-huh", "sure", or any combination of backchannel acknowledgements with fillers (ex. "sure uh uh-huh").

To assign this feature, fifty-two common backchannel acknowledgement tokens are considered. The indicator feature is one (1) if the SU in question is some combination of these backchannel acknowledgements, and zero (0) otherwise.

3.2 SU-level error identification results

We first observe the performance of each feature type in isolation in our maximum entropy framework (Table 1(a)). The top-performing individual

Setup	Features included						F_1-score
	JC04	HPSG	Rules	C-comm	Length	Backchannel	
a) Individual features							
1	√	–	–	–	–	–	79.9
2	–	√	–	–	–	–	77.1
5	–	–	–	–	√	–	59.7
4	–	–	–	√	–	–	42.2
3	–	–	√	–	–	–	23.2
6	–	–	–	–	–	√	0.0
b) All features combined							
7	√	√	√	√	√	√	83.3
c) All-but-one							
8	–	√	√	√	√	√	78.4 (-4.9)
9	√	√	√	–	√	√	81.2 (-2.1)
10	√	–	√	√	√	√	81.3 (-2.0)
11	√	√	–	√	√	√	82.1 (-1.2)
12	√	√	√	√	√	–	82.9 (-0.4)
13	√	√	√	√	–	√	83.2 (-0.1)

Table 1: Comparison of poor construction identification features, tested on the SSR test corpus.

feature is the JC04 edit indicator, which is not surprising as this is the one feature whose existence was designed specifically to predict speaker errors. Following JC04 in individual performance are the HPSG parsability feature, length feature, and unseen c-command rule presence feature. Backchannel acknowledgements had no predictive power on their own. This was itself unsurprising as the feature was primarily meant to reduce the probability of selecting these SUs as errorful.

Combining all rules together (Table 1(b)), we note an F_1-score gain of 3.4 as compared to the top individual feature JC04. (JC04 has a precision of 97.6, recall of 67.6, and F of 79.9; the combined feature model has a precision of 93.0, a recall of 75.3, and an F of 83.3, so unsurprisingly our gain primarily comes from increased error recall).

In order to understand the contribution of an individual feature, it helps not only to see the prediction results conditioned only on that feature, but the loss in accuracy seen when only that feature is removed from the set. We see in Table 1(c) that, though the c-command prediction feature was only moderately accurate in predicting SU errors on its own, it has the second largest impact after JC04 (an F-score loss of 2.1) when removed from the set of features. Such a change indicates the orthogonality of the information within this feature to the other features studied. Length, on the other hand, while moderately powerful as a single indicator, had negligible impact on classification accuracy when removed from the feature set. This indicates that the relationship between errorful sentences and length can be explained away by the other features in our set.

We also note that the combination of all features excluding JC04 is competitive with JC04 itself. Additional complementary features seem likely to further compete with the JC04 prediction feature.

4 Combining efforts

The FHJ work shows that the predictive power of a CRF model could greatly improve (given a restriction on only altering SUs suspected to contain errors) from an F-score of 84.7 to as high as 88.7 for rough copy (RC) errors and from an F-score of 47.5 to as high as 73.8 for non-copy (NC) errors.

Now that we have built a model to predict construction errors on the utterance level, we combine the two approaches to analyze the improvement possible for word-level identification (measured again by precision, recall, and F-score) and for SU-level correction (measured by the SU_Match metric defined in Section 4.2).

4.1 Word-level evaluation of error identification, post SU filtering

We first evaluate edit detection accuracy on those test SUs predicted to be errorful on a per-word basis. To evaluate our progress identifying word-

level error classes, we calculate precision, recall and F-scores for each labeled class c in each experimental scenario. As usual, these metrics are calculated as ratios of correct, false, and missed predictions. However, to take advantage of the double reconstruction annotations provided in SSR (and more importantly, in recognition of the occasional ambiguities of reconstruction) we modified these calculations slightly to account for all references.

Analysis of word-level label evaluation, post SU filtering. Word-level F_1-score results for error region identification are shown in Table 2.

By first automatically selecting testing as described in Section 3 (with a sentence-level F-score of 83.3, Table 1(b)), we see in Table 2 some gain in F-score for all three error classes, though much potential improvement remains based on the oracle gain (rows indicated as having "Gold errors" testing data). Note that there are no results from training only on errorful data but testing on all data, as this was shown to yield dramatically worse results due to data mismatch issues.

Unlike in the experiments where all data was used for testing and training, the best NC and RC detection performance given the automatically selected testing data was achieved when training a CRF model to detect each class separately (RC or NC alone) and not in conjunction with filler word detection FL. As in FHJ, training RC and NC models separately instead of in a joint FL+RC+NC model yielded higher accuracy.

We notice also that the F-score for RC identification is lower when automatically filtering the test data. There are two likely causes. The most likely issue is that the automatic SU-error classifier filtered out some SUs with true RC errors which had previously been correctly identified, reducing the overall precision ratio as well as recall (i.e., we no longer receive accuracy credit for some easier errors once caught). A second, related possibility is that the errorful SUs identified by the Section 3 method had a higher density of errors that the current CRF word-level classification model is unable to identify (i.e. the more difficult errors are now a higher relative percentage of the errors we need to catch). While the former possibility seems more likely, both causes should be investigated in future work.

The F-score gain in NC identification from 42.5 to 54.6 came primarily from a gain in precision (in the original model, many non-errorful SUs were mistakenly determined to include errors). Though capturing approximately 55% of the non-copy NC errors (for SUs likely to have errors) is an improvement, this remains a challenging and unsolved task which should be investigated further in the future.

4.2 Sentence-level evaluation of error identification and region deletion, post SU identification

Depending on the downstream task of speech reconstruction, it may be imperative not only to identify many of the errors in a given spoken utterance, but indeed to identify *all* errors (and only those errors), yielding the exact cleaned sentence that a human annotator might provide.

In these experiments we apply *simple cleanup* (as described in Section 1.1) to both JC04 output and the predicted output for each experimental setup, deleting words when their error class is a filler, rough copy or non-copy.

Taking advantage of the dual annotations provided for each sentence in the SSR corpus, we can report double-reference evaluation. Thus, we judge that if a hypothesized cleaned sentence exactly matches *either* reference sentence cleaned in the same manner we count the cleaned utterance as correct, and otherwise we assign no credit. We report double-reference exact match evaluation between a given SU s and references $r \in R$, as defined below.

$$\text{SU\_match} = \frac{1}{S} \sum_{s \in S} \max_{r \in R} \delta(s, r) \qquad (2)$$

Analysis of sentence level evaluation, post SU identification. Results from this second evaluation of rough copy and non-copy error reconstruction can be seen in Table 3.

As seen in word-level identification results (Table 2), automatically selecting a subset of testing data upon which to apply simple cleanup reconstruction does not perform at the accuracy shown to be possible given an oracle filtering. While measuring improvement is difficult (here, non-filtered data is incomparable to filtered test data results since a majority of these sentences require no major deletions at all), we note again that our methods (MaxEnt/FHJ-x) outperform the baseline of deleting nothing but filled pauses like "eh" and "um", as well as the state-of-the-art baseline JC04.

Class labeled	Training	SUs for Testing	FL	RC	NC
FL+RC+NC	All data	All SU data	71.0	80.3	47.4
	Errorful only	Auto ID'd SU errors	87.9	79.9	49.0
	Errorful only	Gold SU errors	91.6	84.1	52.2
NC	All data	All SU data	-	-	42.5
	Errorful only	Auto ID'd SU errors	-	-	**54.6**
	Errorful only	Gold SU errors	-	-	73.8
NC+FL	All data	All SU data	70.8	-	47.5
	Errorful only	Auto ID'd SU errors	**88.8**	-	53.3
	Errorful only	Gold SU errors	90.7	-	69.8
RC	All data	All SU data	-	*/84.2/*	-
	Errorful only	Auto ID'd SU errors	-	**81.3**	-
	Errorful only	Gold SU errors	-	88.7	-
RC+FL	All data	All SU data	67.8	*/84.7/*	-
	Errorful only	Auto ID'd SU errors	88.1	80.5	-
	Errorful only	Gold SU errors	92.3	87.4	-

Table 2: Error predictions, post-SU identification: F_1-score results. Automatically identified "SUs for testing" were determined via the maximum entropy classification model described earlier in this paper, and feature set #7 from Table 1. Filler (FL), rough copy error (RC) and non-copy error (NC) results are given in terms of word-level F_1-score. **Bold** numbers indicate the highest performance post-automatic filter for each of the three classes. *Italicized* values indicate experiments where no filtering outperformed automatic filtering (for RC errors).

Setup	Classed deleted	Testing	# SUs (filt/unfilt)	# SUs that match ref	% accuracy
Baseline-1	only filled pauses	All data	2288	1800	78.7%
JC04-1	E+FL	All data	2288	1858	81.2%
MaxEnt/FHJ-1	FL+RC+NC	All data	2288	1922	84.0%
Baseline-2	**only filled pauses**	**Auto ID'd**	**430**	**84**	**19.5%**
JC04-2	**E+FL**	**Auto ID'd**	**430**	**187**	**43.5%**
MaxEnt/FHJ-2	**FL+RC+NC**	**Auto ID'd**	**430**	**223**	**51.9%**
Baseline-3	only filled pauses	Gold errors	281	5	1.8%
JC04-3	E+FL	Gold errors	281	126	44.8%
MaxEnt/FHJ-3	FL+RC+NC	Gold errors	281	156	55.5%

Table 3: Error predictions, post-SU identification: Exact Sentence Match Results.
For the baseline, we delete only filled pause filler words like "eh" and "um". For JC04 output, we deleted any word assigned the class E or FL. Finally, for the MaxEnt/FHJ models, we used the jointly trained FL+RC+NC CRF model and deleted all words assigned any of the three classes.

5 Future Work

While some success and improvements for the automatic detection and deletion of fillers and reparanda (i.e., "simple cleanup") have been demonstrated in this work, much remains to be done to adequately address the issues and criteria considered here for full reconstruction of spontaneous speech.

Included features for both the word level and SU-level error detection have only skimmed the surface of potentially powerful features for spontaneous speech reconstruction. There should be continued development of complementary parser-based features (such as those from dependency parsers or even deep syntax parsers such as implementations of HPSG as well as additional syntactic features based on automatic constituent or context-free grammar based parsers). Prosodic

features, though demonstrated to be unnecessary for at least moderately successful detection of simple errors, also hold promise for additional gains. Future investigators should evaluate the gains possible by integrating this information into the features and ideas presented here.

6 Summary and conclusions

This work was an extension of the results in FHJ, which showed that automatically determining which utterances contain errors before attempting to identify and delete fillers and reparanda has the potential to increase accuracy significantly.

In Section 3, we built a maximum entropy classification model to assign binary error classes to spontaneous speech utterances. Six features – JC04, HPSG, unseen rules, unseen c-command relationships, utterance length, and backchannel acknowledgement composition – were considered. The combined model achieved a precision of 93.0, a recall of 75.3, and an F_1-score of 83.3.

We then, in Section 4, cascaded the sentence-level error identification system output into the FHJ word-level error identification and simple cleanup system. This combination lead to non-copy error identification with an F_1-score of 54.6, up from 47.5 in the experiments conducted on all data instead of data identified to be errorful, while maintaining accuracy for rough copy errors and increasing filler detection accuracy as well. Though the data setup is slightly different, the true errors are common across both sets of SUs and thus the results are comparable.

This work demonstrates that automatically selecting a subset of SUs upon which to implement reconstruction improves the accuracy of non-copy (restart fragment) reparanda identification and cleaning, though less improvement results from doing the same for rough copy identification.

Acknowledgments

The authors thank our anonymous reviewers for their valuable comments. Support for this work was provided by NSF PIRE Grant No. OISE-0530118. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the supporting agency.

References

Ellen G. Bard, Robin J. Lickley, and Matthew P. Aylett. 2001. Is disfluency just difficult? In *Disfluencies in Spontaneous Speech Workshop*, pages 97–100.

Eugene Charniak. 1999. A maximum-entropy-inspired parser. In *Proceedings of the Annual Meeting of the North American Association for Computational Linguistics*.

Noam Chomsky, 1970. *Remarks on nominalization*, pages 184–221. Waltham: Ginn.

Christopher Cieri, Stephanie Strassel, Mohamed Maamouri, Shudong Huang, James Fiumara, David Graff, Kevin Walker, and Mark Liberman. 2004. Linguistic resource creation and distribution for EARS. In *Rich Transcription Fall Workshop*.

Hal Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://pub.hal3.name\#daume04cg-bfgs`, implementation available at `http://hal3.name/megam/`, August.

Erin Fitzgerald and Frederick Jelinek. 2008. Linguistic resources for reconstructing spontaneous speech text. In *Proceedings of the Language Resources and Evaluation Conference*.

Erin Fitzgerald, Keith Hall, and Frederick Jelinek. 2009. Reconstructing false start errors in spontaneous speech text. In *Proceedings of the Annual Meeting of the European Association for Computational Linguistics*.

Erin Fitzgerald. 2009. *Reconstructing Spontaneous Speech*. Ph.D. thesis, The Johns Hopkins University.

Dan Flickinger. 2002. On building a more efficient grammar by exploiting types. In Stephan Oepen, Dan Flickinger, Jun'ichi Tsujii, and Hans Uszkoreit, editors, *Collaborative Language Engineering*, pages 1–17. CSLI Publications, Stanford.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 517–520, San Francisco.

Matthias Honal and Tanja Schultz. 2005. Automatic disfluency removal on recognized spontaneous speech – rapid adaptation to speaker-dependent disfluenices. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy channel model of speech repairs. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Jeremy Kahn, Matthew Lease, Eugene Charniak, Mark Johnson, and Mari Ostendorf. 2005. Effective use of prosody in parsing conversational speech. In *Proceedings of the Conference on Human Language Technology*, pages 561–568.

Yang Liu, Elizabeth Shriberg, Andreas Stolcke, Barbara Peskin, and Mary Harper. 2004. The ICSI/UW RT04 structural metadata extraction system. In *Rich Transcription Fall Workshop*.

Sharon L. Oviatt. 1995. Predicting and managing spoken disfluencies during human-computer interaction. *Computer Speech and Language*, 9:19–35.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chiacgo Press and CSLI Publications, Chicago and Stanford.

Elizabeth Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.D. thesis, University of California, Berkeley.

Wolfgang Wahlster, editor. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer, Berlin.

Qi Zhang and Fuliang Weng. 2005. Exploring features for identifying edited regions in disfluent sentences. In *Proceedings of the International Workshop on Parsing Techniques*, pages 179–185.

Yi Zhang, Valia Kordoni, and Erin Fitzgerald. 2007. Partial parse selection for robust deep processing. In *Proceedings of ACL Workshop on Deep Linguistic Processing*, pages 128–135.

Estimating Semantic Distance Using Soft Semantic Constraints in Knowledge-Source–Corpus Hybrid Models

Yuval Marton[*†], **Saif Mohammad**[†], and **Philip Resnik**[*†]
[*]Department of Linguistics and
[†]Laboratory for Computational Linguistics and Information Processing,
Institute for Advanced Computer Studies.
University of Maryland, College Park, MD 20742-7505, USA.
{ymarton,saif,resnik}@umiacs.umd.edu

Abstract

Strictly corpus-based measures of semantic distance conflate co-occurrence information pertaining to the many possible senses of target words. We propose a corpus–thesaurus hybrid method that uses soft constraints to generate word-sense-aware distributional profiles (DPs) from coarser "concept DPs" (derived from a Roget-like thesaurus) and sense-unaware traditional word DPs (derived from raw text). Although it uses a knowledge source, the method is not vocabulary-limited: if the target word is not in the thesaurus, the method falls back gracefully on the word's co-occurrence information. This allows the method to access valuable information encoded in a lexical resource, such as a thesaurus, while still being able to effectively handle domain-specific terms and named entities. Experiments on word-pair ranking by semantic distance show the new hybrid method to be superior to others.

1 Introduction

Semantic distance is a measure of the closeness in meaning of two concepts. People are consistent judges of semantic distance. For example, we can easily tell that the concepts of "exercise" and "jog" are closer in meaning than "exercise" and "theater". Studies asking native speakers of a language to rank word pairs in order of semantic distance confirm this—average inter-annotator correlation on ranking word pairs in order of semantic distance has been repeatedly shown to be around 0.9 (Rubenstein and Goodenough, 1965; Resnik, 1999).

A number of natural language tasks such as machine translation (Lopez, 2008) and word sense disambiguation (Banerjee and Pedersen, 2003; McCarthy, 2006), can be framed as semantic distance problems. Thus, developing automatic measures that are in-line with human notions of semantic distance has received much attention. These automatic approaches to semantic distance rely on manually created lexical resources such as WordNet, large amounts of text corpora, or both.

WordNet-based information content measures have been successful (Hirst and Budanitsky, 2005), but there are significant limitations on their applicability. They can be applied only if a WordNet exists for the language of interest (which is not the case for the "low-density" languages); and even if there is a WordNet, a number of domain-specific terms may not be encoded in it. On the other hand, corpus-based distributional measures of semantic distance, such as cosine and α-skew divergence (Dagan et al., 1999), rely on raw text alone (Weeds et al., 2004; Mohammad, 2008). However, when used to rank word pairs in order of semantic distance or correct real-word spelling errors, they have been shown to perform poorly (Weeds et al., 2004; Mohammad and Hirst, 2006).

Mohammad and Hirst (2006) and Patwardhan and Pedersen (2006) argued that word sense ambiguity is a key reason for the poor performance of traditional distributional measures, and they proposed hybrid approaches that are distributional in nature, but also make use of information in lexical resources such as published thesauri and WordNet. However, both these approaches can be applied to estimate the semantic distance between two terms *only* if both terms exist in the lexical resource they rely on. We know lexical resources tend to have limited vocabulary and a large number of domain-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 775–783,
Singapore, 6-7 August 2009.

specific terms are usually not included.

It should also be noted that similarity values from different distance measures are not comparable (even after normalization to the same scale), that is, a similarity score of .75 as per one distance measure does not correspond to the same semantic distance as a similarity score of .75 from another distance measure.[1] Thus if one uses two independent distance measures, in this case: one resource-reliant and one only corpus-dependent, then these two measures are not comparable (and hence cannot be used in tandem), even if both rely—partially or entirely—on distributional corpus statistics.

We propose a hybrid semantic distance method that *inherently* combines the elements of a resource-reliant measure and a strictly corpus-dependent measure by imposing resource-reliant soft constraints on the corpus-dependent model. We choose the Mohammad and Hirst (2006) method as the resource-reliant method and not one of the WordNet-based measures because, unlike the WordNet-based measures, the Mohammad and Hirst method is distributional in nature and so lends itself immediately for combination with traditional distributional similarity measures. Our new hybrid method combines concept–word co-occurrence information (the Mohammad and Hirst distributional profiles of thesaurus concepts (DPC)) with word–word co-occurrence information, to generate word-sense-biased distributional profiles. The "pure" corpus-based distributional profile (a.k.a. *co-occurrence vector*, or *word association vector*), for some target word u, is biased with soft constraints towards each of the concepts c that list u in the thesaurus, to create a distributional profile that is specific to u in the sense that is most related to the other words listed under c.

Thus, this method can make more fine-grained distinctions than the Mohammad and Hirst method, and yet uses word sense information.[2] Our proposed method falls back gracefully to rely only on word-word co-occurrence information if any of the target terms is not listed in the lexical resource. Experiments on the word-pair ranking task on three different datasets show that the our proposed hybrid measure outperforms all other comparable distance measures.

Mohammad and Hirst (2007) show that their method can be used to compute semantic distance in a resource poor language L_1 by combining its text with a thesaurus in a resource-rich language L_2 using an L_1–L_2 bilingual lexicon to create cross-lingual distributional profiles of concepts, that is, L_2 word co-occurrence profiles of L_1 thesaurus concepts. Since our method makes use of the Mohammad and Hirst DPCs, it can just as well make use of their cross-lingual DPCs, to compute semantic distance in a resource-poor language, just as they did. We leave that for future work.

2 Background and Related Work

Strictly speaking, semantic distance/closeness is a property of lexical units—a combination of the surface form and word sense.[3] Two terms are considered to be semantically close if there is a lexical semantic relation between them. Such a relation may be a classical relation such as hypernymy, troponymy, meronymy, and antonymy, or it may be what have been called an ad-hoc non-classical relation, such as cause-and-effect (Morris and Hirst, 2004). If the closeness in meaning is due to certain specific classical relations such as hypernymy and troponymy, then the terms are said to be semantically *similar*. Semantic *relatedness* is the term used to describe the more general form of semantic closeness caused by any semantic relation (Hirst and Budanitsky, 2005). So the nouns *liquid* and *water* are both semantically similar and semantically related, whereas the nouns *boat* and *rudder* are semantically related, but not similar.

The next three sub-sections describe three kinds of automatic distance measures: (1) lexical-resource-based measures that rely on a manually created resource such as WordNet; (2) corpus-based measures that rely only on co-occurrence statistics from large corpora; and (3) hybrid measures that are distributional in nature, and that also exploit the information in a lexical resource.

2.1 Lexical-resource-based measures

WordNet is a manually-created hierarchical network of nodes (taxonomy), where each node in

[1] All we can infer is that if w_1 and w_2 have a similarity score of .75 and w_3 and w_4 have a score of .5 by the *same* distance measure, then w_1–w_2 are closer in meaning than w_3–w_4.

[2] Even though Mohammad and Hirst (2006) use thesaurus categories as coarse concepts, their algorithm can be applied using more finer-grained thesaurus word groupings (paragraphs and semicolon units), as well.

[3] The notion of semantic distance can be generalized, of course, to larger units such as phrases, sentences, passages, and so on (Landauer et al., 1998).

the network represents a fine-grained concept or word sense. An edge between two nodes represents a lexical semantic relation such as hypernymy and troponymy. WordNet-based measures consider two terms to be close if they occur close to each other in the network (connected by only a few arcs), if their definitions share many terms (Banerjee and Pedersen, 2003; Patwardhan and Pedersen, 2006), or if they share a lot of information (Lin, 1998; Resnik, 1999). The length of each arc/link (distance between nodes) can be assumed a unit length, or can be computed from corpus statistics. Within WordNet, the *is-a* hierarchy is much more well-developed than that of other lexical semantic relations. So, not surprisingly, the best WordNet-based measures are those that rely only on the *is-a* hierarchy. Therefore, they are good at measuring semantic similarity (e.g., *doctor–physician*), but not semantic relatedness (e.g., *doctor–scalpel*). Further, the measures can only be used in languages that have a (sufficiently developed) WordNet. WordNet sense information has been criticized to be too fine grained (Agirre and Lopez de Lacalle Lekuona, 2003; Navigli, 2006). See Hirst and Budanitsky (2005) for a comprehensive survey of WordNet-based measures.

2.2 Corpus-based measures

Strictly corpus-based measures of distributional similarity rely on the hypothesis that words that occur in similar context tend to be semantically close (Firth, 1957; Harris, 1940). The set of contexts of each target word u is represented by its distributional profile (DP)—the set of words that tend to co-occur with u within a certain distance, along with numeric scores signifying this co-occurrence tendency with u. Then measures such as cosine or α-skew divergence are used to determine how close the DPs of the two target words are. See Section 3 for more details and related work. These measures are very appealing because they rely simply on raw text, but, as described earlier, when used to rank word pairs in order of semantic distance, or to correct real-word spelling errors, they perform poorly, compared to the WordNet-based measures. See Weeds et al. (2004), Mohammad (2008), and Curran (2004) for detailed surveys of distributional measures.

As Mohammad and Hirst (2006) point out, the DP of a word u conflates information about the potentially many senses of u. For example, consider the following. The noun *bank* has two senses "river bank" and "financial institution". Assume that *bank*, when used in the "financial institution" sense, co-occurred with the noun *money* 100 times in a corpus. Similarly, assume that *bank*, when used in the "river bank" sense, co-occurred with the noun *boat* 80 times. So the DP of *bank* will have co-occurrence information with *money* as well as *boat*:

DPW(*bank*):
money,100; *boat*,80; *bond*,70; *fish*,77; ...

Assume that the DP of the word *ATM* is:

DPW(*ATM*):
money,120; *boat*,0; *bond*,90; *fish*,0; ...

Thus the distributional distance of *bank* with *ATM* will be some sort of an average of the semantic distance between the "financial institution" and "ATM" senses and the semantic distance between the "river bank" and "ATM" senses. However, in various natural language tasks, we need the semantic distance between the intended senses of *bank* and *ATM*, which often also tends to be the semantic distance between their closest senses.

2.3 Hybrid measures

Both Mohammad and Hirst (2006) and Patwardhan and Pedersen (2006) proposed measures that are not only distributional in nature but also rely on a lexical resource to exploit the manually encoded information therein as well as to overcome the sense-conflation problem (described in section 2.2). Since we essentially combine the Mohammad and Hirst method with a "pure" word-based distributional measure to create our hybrid approach, we briefly describe their method here.

Mohammad and Hirst (2006) generate separate distributional profiles for the different senses of a word, without using any sense-annotated data. They use the categories in a Roget-style thesaurus (*Macquaries* (Bernard, 1986)) as coarse senses or concepts. There are about 1000 categories in a thesaurus, and each category has on average 120 closely related words. A word may be found in more than one category if it has multiple meaning. They use a simple unsupervised algorithm to determine the vector of words that tend to co-occur with each concept and the corresponding strength of association (a measure of how strong the tendency to co-occur is). The target word u will be assigned one DPC for each of the concepts that

list u. Below are example DPCs of the two concepts pertaining to *bank*:[4]

DPC("fin. inst."):
money,1000; *boat*,32; *bond*,705; *fish*,0; ...
DPC("river bank"):
money,5; *boat*,863; *bond*,0; *fish*,948; ...

The distance between two words u, v is determined by calculating the closeness of each of the DPCs of u to each of DPCs of v, and the closest DPC-pair distance is chosen.

Mohammad and Hirst (2006) show that their approach performs better than other strictly corpus-based approaches that they experimented with. However, all those experiments were on word-pairs that were listed in the thesaurus. Their approach is not applicable otherwise. In Sections 3 and 4 we show how cosine–log-likelihood-ratio (or any comparable distributional measure) can be combined with the Mohammad and Hirst DPCs to form a hybrid approach that is not limited to the vocabulary of a lexical resource.

Erk and Padó (2008) proposed a way of representing a word sense in context by biasing the target word's DP according to the context surrounding a target (specific) occurrence of the target word. They use dependency relations and selectional preferences of the target word and combine multiple DPs of words appearing in the context of the target occurrence, in a manner so as to give more weight to words co-occurring with both the target word and the target occurrence's context words. The advantage of this approach is that it does not rely on a thesaurus or WordNet. Its disadvantage is that it relies on dependency relations and selectional preferences information, and that the context information it uses in order to determine the word sense is quite limited (only the words surrounding a single occurrence of the and hence the representation of that sense might not be sufficiently accurate. Their approach effectively assumes that each occurrence of a word has a unique sense.

3 Distributional Measures with Soft Semantic Constraints

Traditional distributional profiles of words (DPW) give word–word co-occurrence frequencies. For example, DPW(u) gives the number of times the target word u co-occurs with with all other words:[5]

$\mathrm{DPW}(u)$:
w_1,f(u,w_1); w_2,f(u,w_2); w_3,f(u,w_3); ...

where f stands for co-occurrence frequency (and can be generalized to stand for any strength of association (SoA) measure such as the log-likelihood ratio). Mohammad and Hirst create concept–word co-occurrence vectors, "distributional profiles of concepts" (DPCs), from non-annotated corpus. For example, DPC(c) gives the number of times the concept (thesaurus category) c co-occurs with all the words in a corpus.

$\mathrm{DPC}(c)$:
w_1,f(c,w_1); w_2,f(c,w_2); w_3,f(c,w_3); ...

A target word u that appears under thesaurus concepts c_1, ..., c_n would be assigned to DPC(c_1), ..., DPC(c_n). Therefore, if a target word v also appears under some same concept c, the DPCs of u and v would be indistinguishable.

We propose the creation of distributional profiles of word senses (DPWS(u_c)) that approximate the SoA of the target word u, when used in sense c, with each of the words in the corpus:

$\mathrm{DPWS}(u_c)$:
w_1,f(u_c,w_1); w_2,f(u_c,w_2); w_3,f(u_c,w_3); ...

In order to get exact counts, one needs sense-annotated data. However, such data is expensive to create, and is scarce. Therefore, we propose estimating these counts from the DPW and DPC counts:

$$f(u_c, w_i) = p(c|w_i) \times f(u, w_i) \qquad (1)$$

where the conditional probability $p(c|w_i)$ is calculated from the co-occurrence frequencies in DPCs; and the co-occurrence count $f(u, w_i)$ is calculated from DPWs. If the target word is not in the thesaurus's vocabulary, then we assume uniform distribution over all concepts, and in practice use a single sense, and take the conditional probability to be 1. Since the method takes sense-proportional co-occurrence counts, we will refer to this method as the **hybrid-sense-proportional-counts method** (or, **hybrid-prop** for short).

[4]The relatively large co-occurrence frequency values for DPCs as compared to DPWs is because a concept can be refered to by many words (on average 100).

[5]The dimensions of the DP co-occurrence vector can be defined arbitrarily, and do not have to correspond to the words in the vocabulary. The most notable alternative representation is the Latent Semantic Analysis and its variants (Landauer et al., 1998; Finkelstein et al., 2002; Budiu et al., 2006).

For example, below is the DPWS of *bank* in the "financial institution" sense, calculated from its DPW and DPCs:

DPW(*bank*):
money,100; *boat*,80; *bond*,70; *fish*,77; ...

DPC("fin. inst."):
money,1000; *boat*,32; *bond*,705; *fish*,0; ...
DPC("river bank"):
money,5; *boat*,863; *bond*,0; *fish*,948; ...

DPWS($bank_{\text{"fin.inst."}}$):
money,$(\frac{1000}{1000+5} \times 100)$; *boat*,$(\frac{32}{32+863} \times 80)$;
bond,$(\frac{705}{705+0} \times 70)$; *fish*,$(\frac{0}{0+948} \times 77)$; ...

Once the DPWS are calculated, any counts-based SoA and distance measures can be applied. For example, in this work we use log-likelihood ratio (Dunning, 1993) to determine the SoA between a word sense and co-occurring words, and cosine to determine the distance between two DPWS's log likelihood vectors (McDonald, 2000). We also contrast this measure with cosine of conditional probabilities vectors. Given two target words, we determine the distance between each of their DPWS pairings and the closest DPWS-pair distance is chosen.

3.1 The hybrid-sense-filtered-counts method

Since the DPCs are created in an unsupervised manner, they are expected to be somewhat noisy. Therefore, we also experimented with a variant of the method proposed above, that simply makes use of whether the conditional probability $p(c|w_i)$ is greater than 0 or not:

$$f(u_c, w_i) = \begin{cases} f(u, w_i) & \text{If } p(c|w_i) > 0 \\ 0 & \text{Otherwise} \end{cases} \quad (2)$$

Since this method essentially filters out collocates that are likely not relevant to the target sense c of the target word u, we will refer to this method as the **hybrid-sense-filtered-counts method** (or, just **hybrid-filt** for short). Below is an example hybrid-filtered DPWS of *bank* in the "financial institution" sense:

DPWS($bank_{\text{"fin.inst."}}$:
money,100); *boat*,80; *bond*,70; ...

Note that the collocate *fish* is now filtered out, whereas *bank's* co-occurrence counts with *money, boat,* and *bond* are left as is (and not sense-proportionally attenuated).

4 Evaluation

We evaluated various methods on the task of ranking word pairs in order of semantic distance. These methods included our sense-biased methods as well as several baselines: the Mohammad and Hirst (2006) DPC-based methods, the traditional word-based distributional similarity methods, and several Latent Semantic Analysis (LSA)-based methods. We used three testsets and their corresponding human judgment gold standards: (1) the Rubenstein and Goodenough (1965) set of 65 noun pairs—denoted **RG-65**; (2) the WordSimilarity-353 (Finkelstein et al., 2002) set of 353 noun pairs (which include the RG-65 pairs) of which we discarded of one repeating pair—denoted **WS-353**; and (3) the Resnik and Diab (2000) set of 27 verb pairs—denoted **RD-00**.

4.1 Corpora and Pre-processing

We generated distributional profiles (DPWs and DPCs) from the *British National Corpus (BNC) (Burnard, 2000)*, which is a balanced corpus. We lowercased the characters, and stripped numbers, punctuation marks, and any SGML-like syntactic tags, but kept sentence boundary markers. The BNC contained 102,100,114 tokens of 546,299 types (vocabulary size) after tokenization. For the verb set, we also lemmatized this corpus.

We considered two words as co-occurring if they occurred in a window of ± 5 words from each other. We stoplisted words that co-occurred with more than 2000 word types.

4.2 Results

The Spearman rank correlations of the automatic rankings of the RG-65, WS353, and RD-00 testsets with the corresponding gold-standard human rankings is listed in Table 1.[6] The higher the Spearman rank correlation, the more accurate is the distance measure.

4.2.1 Results on the RG-65 testset

Baselines. We replicated the traditional word-based distributional distance measure using cosine of vectors (DPs) containing conditional probabilities (**word-cos-cp**). Its rank correlation of .53 is close to the correlation of .54 reported in Mohammad and Hirst (2006), hereafter MH06. We replicated the MH06 concept-based approach

[6] Certain experiments were not pursued as they were redundant in supporting our claims.

Method	RG-65	WS-353	RD-00
Baselines (replicated):			
Traditional distributional measures			
word-cos-cp	.53	.31	.46
word-cos-ll	.70	**.54**	.51
word-cos-pmi	.62	.43	.57
Mohammad and Hirst methods and variants			
concept-cos-cp	.62	.38	.41
concept*-cos-cp	.65	.33	.43
concept-cos-ll	.60	.37	.43
concept*-cos-ll	.64	.25	.27
concept*-cos-pmi	.40	.19	.28
Other (LSA and variants)			
LSA	.56	.47	.55
GLSA-cos-pmi	.18	n.p.	n.p.
GLSA-cos-ll	.47	n.p.	.29
New methods:			
hybrid-prop-cos-ll	.72	.49	.53
hybrid-prop*-cos-ll	.69	.46	.45
hybrid-filt-cos-ll	.73	**.54**	.38
hybrid-filt*-cos-ll	**.77**	**.54**	.39
hybrid-prop*-cos-pmi	.58	.43	**.71**
hybrid-filt*-cos-pmi	.61	.42	.64

Table 1: Spearman rank correlation on RG-65, WS-353, and RD-00 testsets, trained on BNC. '*' indicates the use of a smaller bootstrapped concept–word co-occurrence matrix. 'n.p.' indicates that the experiment was not pursued.

(**concept-cos-cp**), and its bootstrapped variant that uses a smaller concept–word co-occurrence matrix (**concept*-cos-cp**). The latter yielded a correlation score .65, close to the .69 reported in MH06.

We also experimented with cosine of PMI vectors (**word-cos-pmi**) which obtained a correlation of .62. Log likelihood ratios (**word-cos-ll**) gave best results among the baseline methods (.70), and so we it more often in the implementations of our hybrid method.

We conducted experiments with LSA and its GLSA variants (Budiu et al., 2006) as additional baselines. A limited vocabulary of the 33,000 most frequent words in the BNC and all test words was used in these experiments. (A larger vocabulary was computationally expensive and 33,000 is also the vocabulary size used by Budiu et al. (2006) in their LSA experiments.)

New Methods: The hybrid method variants proposed in this paper (**hybrid-prop-cos-ll** and **hybrid-filt-cos-ll**) were the best performers on the RG-65 test set. Particularly, they performed better than both the traditional word-distance measures (**word-cos-ll**), and our concept-based methods—variants of the MH06 method that are used with likelihood ratios (**concept-cos-ll, concept*-cos-ll**). The -pmi methods were all poorer performers than their -ll counterparts. The -pmi hybrid variants obtained higher scores than the concept-based ones, but almost the same scores as the word-based ones.

4.2.2 Results on WS-353 and RD-00 testsets

On WS-353, all our hybrid methods out-performed their concept counterparts, and were on par with their word-based counterparts. On RD-00, **word-cos-pmi** out-performed all other word-based methods, and the hybrid -pmi methods were best performers with scores of .64 and .71. Our word-cos-ll, hybrid-prop-cos-ll, and the two hybrid pmi results on RD-00 are better than any non-WordNet results reported by Resnik and Diab (2000), including their syntax-informed methods—the variants of Lin ("distrib", .43) and Dorr ("LCS", .39). In fact, our hybrid*-prop-cos-pmi and hybrid*-filt-cos-pmi results reach correlation levels of the WordNet-based methods reported there (.66–.68). Also, on WS-353, our hybrid sense-filtered variants and word-cos-ll obtained a correlation score higher than published results using WordNet-based measures (Jarmasz and Szpakowicz, 2003) (.33 to .35) and Wikipedia-based methods (Ponzetto and Strube, 2006) (.19 to .48); and very close to the results obtained by thesaurus-based (Jarmasz and Szpakowicz, 2003) (.55) and LSA-based methods (Finkelstein et al., 2002) (.56).

The lower correlation scores of all measures on the WS-353 test set are possibly due to it having politically biased word pairs (examples include: *Arafat–peace, Arafat–terror, Jerusalem–Palestinian*) for which BNC texts are likely to induce low correlation with the human raters of WS-353. This testset also has disproportionately many terms from the news domain.

The concept methods performed poorly on WS-353 partly because many of the target words do not exist in the thesaurus. For instance, there were 17 such word types that occurred in 20 WS-353 testset word pairs. When excluding these pairs, concept-cos-cp goes up from .38 to .45, and concept*-cos-pmi from .19 to .24. Interestingly, results of the hybrid methods show that they were largely unaffected by the out-of-vocabulary problem on the WS-353 dataset.

On the verbs dataset RD-00, while hybrid-prop-cos-ll fared slightly better than word-cos-ll, using the smaller matrix seemed to hurt performance of

hybrid*-prop-cos-ll compared to word-cos-ll. But results suggest that the -pmi methods might serve as a better measure than -ll for verbs, although this claim should be tested more rigorously.

Human judgments of semantic distance are less consistent on verb-pairs than on noun-pairs, as reflected in inter-rater agreement measures in Resnik and Diab (2000) and others). Thus, not surprisingly, the scores of almost all measures are lower for the verb data than the RG-65 noun data.

5 Discussion

The hybrid methods proposed in this paper obtained higher accuracies than all other methods on the RG-65 testset (all of whose words were in the published thesaurus), and on the RD-00 testset, and their performance was at least respectable on the WS-353 testset (many of whose words were not in the published thesaurus). This is in contrast to the concept-distance methods which suffer greatly when the target words are not in the lexical resource (here, the thesaurus) they rely on, even though these methods can make use of co-occurrence information of words not in the thesaurus with concepts from the thesaurus.

Amongst the two hybrid methods proposed, the **sense-filtered-counts** method performed better using the smaller bootstrapped concept–word co-occurrence matrix whereas the sense-proportional method performed better using the larger concept–word co-occurrence matrix. We believe this is because the bootstrapping method proposed in Mohammad and Hirst (2006) has the effect of resetting to 0 the small co-occurrence counts. The noise from these small co-occurrence counts affects the sense-filtered-counts method more adversely (since any non-zero value will cause the inclusion of the corresponding collocate's full co-occurrence count) and so the bootstrapped matrix is more suitable for this method.

The results also show that the cosine of log-likelihood ratios method mostly performs better than cosine of conditional probabilities and the pmi methods on the noun sets. This further supports the claim by Dunning (1993) that log-likelihood ratio is much less sensitive than pmi to low counts. Interestingly, on the verb set, the pmi methods, and especially hybrid*-prop-cos-pmi, did extremely well. Further investigation is needed in order to determine if pmi is indeed more suitable for verb semantic similarity, and why.

6 Conclusion

Traditional distributional similarity conflates co-occurrence information pertaining to the many senses of the target words. Mohammad and Hirst (2006) show how distributional measures can be used to compute distance between very coarse word senses or concepts (thesaurus categories), and even obtain better results than traditional distributional similarity. However, their method requires that the target words be listed in the thesaurus, which is often not the case for domain-specific terms and named entities. In this paper, we proposed hybrid methods (**hybrid-sense-filtered-counts** and **hybrid-sense-proportional-counts**) that combine word–word co-occurrence information (traditional distributional similarity) with word–concept co-occurrence information (Mohammad and Hirst, 2006), with soft constraints in such a manner that the method makes use of information encoded in the thesaurus when available, and degrades gracefully if the target word is not listed in the thesaurus. Our method generates word-sense-biased distributional profiles (DPs) from non-annotated corpus-based word-based DPs and coarser-grained aggregated thesaurus-based "concept DPs" (DPCs). We showed that the hybrid method correlates with human judgments of semantic distance in most cases better than any of the other methods we replicated.

We are now interested in improving semantic distance measures for verb–verb, adjective–adjective, and cross-part-of-speech pairs, by exploiting specific information pertaining to these parts of speech in lexical resources in addition to purely co-occurrence information.

Acknowledgments

We thank Mona Diab for her help with her verb test set, Raluca Budiu for her help and clarifications regarding the GLSA method and its implementation details, and the anonymous reviewers for their valuable feedback. This work was supported, in part, by the National Science Foundation under Grant No. IIS-0705832, and in part, by the Human Language Technology Center of Excellence. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsor.

References

Eneko Agirre and Oier Lopez de Lacalle Lekuona. 2003. Clustering WordNet word senses. In *Proceedings of the 1st International Conference on Recent Advances in Natural Language Processing (RANLP-2003)*, Borovets, Bulgaria.

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 805–810, Acapulco, Mexico.

John R. L. Bernard, editor. 1986. *The Macquarie Thesaurus*. Macquarie Library, Sydney, Australia.

Raluca Budiu, Christiaan Royer, and Peter Pirolli. 2006. Modeling information scent: A comparison of LSA, PMI and GLSA similarity measures on common tests and corpora. In *Proceedings of RIAO'07*, Pittsburgh, PA.

Lou Burnard. 2000. *Reference Guide for the British National Corpus*. Oxford University Computing Services, Oxford, England, world edition edition.

James R. Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, School of Informatics, University of Edinburgh, Edinburgh, UK.

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

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2086)*, pages 897–906, Honolulu, HI.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

John R. Firth. 1957. A synopsis of linguistic theory 193055. *Studies in Linguistic Analysis*, (special volume of the Philological Society):132. Distributional Hypothesis.

Zellig S. Harris. 1940. Review of Louis H. Gray, foundations of language (New York: Macmillan, 1939). *Language*, 16(3):216–231.

Graeme Hirst and Alexander Budanitsky. 2005. Correcting real-word spelling errors by restoring lexical cohesion. *NLE*, 11(1):87–111.

Mario Jarmasz and Stan Szpakowicz. 2003. Roget's Thesaurus and semantic similarity. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-2003)*, pages 212–219, Borovets, Bulgaria.

Thomas Landauer, Peter Foltz, and Darrell Laham. 1998. Introduction to latent semantic analysis. *Discourse Processes*, 25:259 – 284.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, page 296304, San Francisco, CA.

Adam Lopez. 2008. Statistical machine translation. *ACM Computing Surveys*, 40(3):149.

Diana McCarthy. 2006. Relating WordNet senses for word sense disambiguation. In *Proceedings of the European Chapter of the Association for Computational Linguistics Workshop Making Sense of Sense - Bringing Computational Linguistics and Psycholinguistics Together*, pages 17–24, Trento, Italy.

S. McDonald. 2000. *Environmental determinants of lexical processing effort*. Ph.D. thesis, University of Edinburgh, Edinburgh, UK.

Saif Mohammad and Graeme Hirst. 2006. Distributional measures of concept-distance: A task-oriented evaluation. In *Proceedings of EMNLP*.

Saif Mohammad, Iryna Gurevych, Graeme Hirst, and Torsten Zesch. 2007. Cross-lingual distributional profiles of concepts for measuring semantic distance. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL-2007)*, pages 571–580, Prague, Czech Republic.

Saif Mohammad. 2008. *Measuring Semantic Distance using Distributional Profiles of Concepts*. Ph.D. thesis, Department of Computer Science, University of Toronto, Toronto, Canada.

Jane Morris and Graeme Hirst. 2004. Non-classical lexical semantic relations. In *Proceedings of the Workshop on Computational Lexical Semantics, Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 46–51, Boston, Massachusetts.

Roberto Navigli. 2006. Meaningful clustering of senses helps boost word sense disambiguation performance. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association*, pages 105–112, Sydney, Australia.

Siddharth Patwardhan and Ted Pedersen. 2006. Using WordNet based context vectors to estimate the semantic relatedness of concepts. In *Proceedings of Making Sense of Sense EACL Workshop*, pages 1–8.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-2006)*, pages 192–199, New York, NY.

Philip Resnik and Mona Diab. 2000. Measuring verb similarity. In *22nd Annual Meeting of the Cognitive Science Society (COGSCI2000)*, Philadelphia, PA.

Philip Resnik. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *JAIR*, 11:95–130.

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

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, pages 1015–1021, Geneva, Switzerland.

Recognizing Textual Relatedness with Predicate-Argument Structures

Rui Wang
Dept of Computational Linguistics
Saarland University
66123 Saarbrücken, Germany
rwang@coli.uni-sb.de

Yi Zhang
Dept of Computational Linguistics
Saarland University
LT-Lab, DFKI GmbH
D-66123 Saarbrücken, Germany
yzhang@coli.uni-sb.de

Abstract

In this paper, we first compare several strategies to handle the newly proposed three-way *Recognizing Textual Entailment* (RTE) task. Then we define a new measurement for a pair of texts, called *Textual Relatedness*, which is a weaker concept than semantic similarity or paraphrase. We show that an alignment model based on the predicate-argument structures using this measurement can help an RTE system to recognize the *Unknown* cases at the first stage, and contribute to the improvement of the overall performance in the RTE task. In addition, several heterogeneous lexical resources are tested, and different contributions from them are observed.

1 Introduction

Recognizing Textual Entailment (RTE) (Dagan et al., 2006) is a task to detect whether one *Hypothesis* (H) can be inferred (or entailed) by a *Text* (T). Being a challenging task, it has been shown that it is helpful to applications like question answering (Harabagiu and Hickl, 2006). The recent research on RTE extends the two-way annotation into three-way[1] [2], making it even more difficult, but more linguistic-motivated.

The straightforward strategy is to treat it as a three-way classification task, but the performance suffers a significant drop even when using the same classifier and the same feature model. In fact, it can also be dealt with as an extension to the traditional two-way classification, e.g., by identifying the *Entailment* (E) cases first and then further label the *Contradiction* (C) and *Unknown* (U) T-H pairs. Some other researchers also work on detecting negative cases, i.e. contradiction, instead of entailment (de Marneffe et al., 2008). However, according to our best knowledge, the detailed comparison between these strategies has not been fully explored, let alone the impact of the linguistic motivation behind the strategy selection. This paper will address this issue.

Take the following example from the RTE-4 test set (Giampiccolo et al., 2009) as an example,

T: *At least five people have been killed in a head-on train collision in north-eastern France, while others are still trapped in the wreckage. All the victims are adults.*

H: *A French train crash killed children.*

This is a pair of two contradicting texts, the mentioning of events (i.e. train crash) in both T and H are assumed to refer the same event[3]. In fact, the only contradicting part lies in the second sentence of T against H, that is, whether there are children among the victims. Therefore, this pair could also be classified as a *Known* (K) pair (=E∪C) against *Unknown* (U) pairs, instead of being classified as a *Non-entailment* (N) case (=C∪U) against E case in the traditional two-way annotation.

Furthermore, many state-of-the-art RTE approaches which are based on overlapping information or similarity functions between T and H, in fact over-cover the E cases, and sometimes, cover the C cases as well. Therefore, in this paper, we

[1] http://nlp.stanford.edu/RTE3-pilot/

[2] http://www.nist.gov/tac/tracks/2008/rte/rte.08.guidelines.html

[3] See more details about the annotation guideline at http://www.nist.gov/tac/tracks/2008/rte/rte.08.guidelines.html

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 784–792,
Singapore, 6-7 August 2009.

would like to test whether applying this style of approaches to capture the K cases instead of E cases is more effective. While in lexical semantics, semantic relatedness is a weaker concept than semantic similarity, there is no counterpart at the sentence or text level. Therefore, in this paper, we propose a *Recognizing Textual Relatedness (RTR)* task as a subtask or the first step of RTE. By doing so, we choose *predicate-argument structure (PAS)* as the feature representation, which has already been shown quite useful in the previous RTE challenges (Wang and Neumann, 2007).

In order to obtain the PAS, we utilize a Semantic Role Labeling (SRL) system developed by Zhang et al. (2008). Although SRL has been shown to be effective for many tasks, e.g. information extraction, question answering, etc., it has not been successfully used for RTE, mainly due to the low coverage of the verb frame or semantic role resources or the low performance of the automatic SRL systems. The recent CoNLL shared tasks (Surdeanu et al., 2008; Hajič et al., 2009) have been focusing on semantic dependency parsing along with the traditional syntactic dependency parsing. The PAS from the system output is almost ready for use to build applications based on it. Therefore, another focus of this paper will be to apply SRL to the RTE task. In particular, it can improve the first stage binary classification (K vs. U), and the final result improves as well.

The rest of the paper will be organized as follows: Section 2 will give a brief literature review on both RTE and SRL; Section 3 describes the semantic parsing system, which includes a syntactic dependency parser and an SRL system; Section 4 presents an algorithm to align two PASs to recognize textual relatedness between T and H, using several lexical resources; The experiments will be described in Section 5, followed by discussions; and the final section will conclude the paper and point out directions to work on in the future.

2 Related Work

Although the term of *Textual Relatedness* has not been widely used by the community (as far as we know), many researchers have already incorporated modules to tackle it, which are usually implemented as an alignment module before the inference/learning module is applied. For example, Pado et al. (2009) mentioned two alignment modules, one is a phrase-based alignment system called MANLI (MacCartney et al., 2008), and the other is a stochastic aligner based on dependency graphs. Siblini and Kosseim (2009) performed the alignment on top of two ontologies. In this paper, we would like to follow this line of research but on another level of representation, i.e. the predicate-argument structures (PAS), together with different lexical semantic resources.

As for the whole RTE task, many people directly do the three-way classification with selective features (e.g. Agichtein et al. (2009)) or different inference rules to identify entailment and contradiction simultaneously (e.g. Clark and Harrison (2009)); while other researchers also extend their two-way classification system into three-way by performing a second-stage classification afterwards. An interesting task proposed by de Marneffe et al. (2008) suggested an alternative way to deal with the three-way classification, that is, to split out the contradiction cases first. However, it has been shown to be more difficult than the entailment recognition. Based on these previous works and our experimental observations, we propose an alternative two-stage binary classification approach, i.e. to identify the unknown cases from the known cases (entailment and contradiction) first. And the results show that due to the nature of these approaches based on overlapping information or similarity between T and H, this way of splitting is more reasonable.

However, RTE systems using semantic role labelers has not shown very promising results, although SRL has been successfully used in many other NLP tasks, e.g. information extraction, question answering, etc. According to our analysis of the data, there are mainly three reasons: a) the limited coverage of the verb frames or predicates; b) the undetermined relationships between two frames or predicates; and c) the unsatisfying performance of an automatic SRL system. For instance, Burchardt et al. (2007) attempted to use FrameNet (Baker et al., 1998) for the RTE-3 challenge, but did not show substantial improvement. With the recent CoNLL challenges, more and more robust and accurate SRL systems are ready for use, especially for the PAS identification. For the lexical semantics, we also discover that, if we relax the matching criteria (from similarity to relatedness), heterougeous resources can contribute to the coverage differently and then the effectiveness of PAS will be shown as well.

3 Semantic Parsing

In order to obtain the predicate-argument structures for the textual entailment corpus, we use the semantic role labeler described in (Zhang et al., 2008). The SRL system is trained on the Wall Street Journal sections of the Penn Treebank using PropBank and NomBank annotation of verbal and nominal predicates, and relations to their arguments, and produces as outputs the semantic dependencies. The head words of the arguments (including modifiers) are annotated as a direct dependent of the corresponding predicate words, labeled with the type of the semantic relation (Arg0, Arg1 ..., and various ArgMs). Note that for the application of SRL in RTE task, the PropBank and NomBank notation appears to be more accessible and robust than the the FrameNet notation (with much more detailed roles or frame elements bond to specific verb frames).

As input, the SRL system requires syntactic dependency analysis. We use the open source MST Parser (McDonald et al., 2005), trained also on the Wall Street Journal Sections of the Penn Treebank, using a projective decoder with second-order features. Then the SRL system goes through a pipeline of 4-stage processing: predicate identification (PI) identifies words that evokes a semantic predicate; argument identification (AI) identifies the arguments of the predicates; argument classification (AC) labels the argument with the semantic relations (roles); and predicate classification (PC) further differentiate different use of the predicate word. All components are built as maximal entropy based classifiers, with their parameters estimated by the open source TADM system[4], feature sets selected on the development set. Evaluation results from previous years' CoNLL shared tasks show that the system achieves state-of-the-art performance, especially for its out-domain applications.

4 Textual Relatedness

As we mentioned in the introduction, we break down the three-way classification into a two-stage binary classification. Furthermore, we treat the first stage as a subtask of the main task, which determines whether H is relevant to T. Similar to the probabilistic entailment score, we use a relatedness score to measure such relationship. Due to the nature of the entailment recognition that H should be fully entailed by T, we also make this relatedness relationship asymmetric. Roughly speaking, this *Relatedness* function $R(T, H)$ can be described as whether or how relevant H is to some part of T. The relevance can be realized as string similarity, semantic similarity, or being co-occurred in similar contexts.

Before we define the relatedness function formally, let us look at the representation again. After semantic parsing described in the previous section, we obtain a PAS for each sentence. On top of it, we define a *predicate-argument graph* (PAG), the nodes of which are predicates, arguments or sometimes both, and the edges of which are labeled semantic relations. Notice that each predicate can dominate zero, one, or more arguments, and each argument have one or more predicates which dominate it. Furthermore, the graph is not necessarily fully connected. Thus, the $R(T, H)$ function can be defined on the dependency representation as follows: if the PAG of H is semantically relevant to part of the PAG of T, H is semantically relevant to T.

In order to compare the two graphs, we further reduce the alignment complexity by breaking the graphs into sets of trees. Two types of decomposed trees are considered: one is to take each predicate as the root of a tree and arguments as child nodes, and the other is on the contrary, to take each argument as root and their governing predicates as child nodes. We name them as *Predicate Trees* (P-Trees) and *Argument Trees* (A-Trees) respectively. To obtain the P-Trees, we enumerate each predicate, find all the arguments which it directly dominates, and then construct a P-Tree. The algorithm to obtain A-Trees works in the similar way. Finally, we will have a set of P-Trees and a set of A-Trees for each PAG, both of which are simple trees with depth of one. Figure 1 shows an example of such procedures. Notice that we do not consider cross-sentential inference, instead, we simply take the union of tree sets from all the sentences. Figure 2 illustrates the PAG for both T and H after semantic parsing, and the resulting P-Trees and A-Trees after applying the decomposition algorithm.

Formally, we define the relatedness function for a T-H pair as the maximum value of the relatedness scores of all pairs of trees in T and H (P-trees and A-trees).

[4] http://tadm.sourceforge.net/

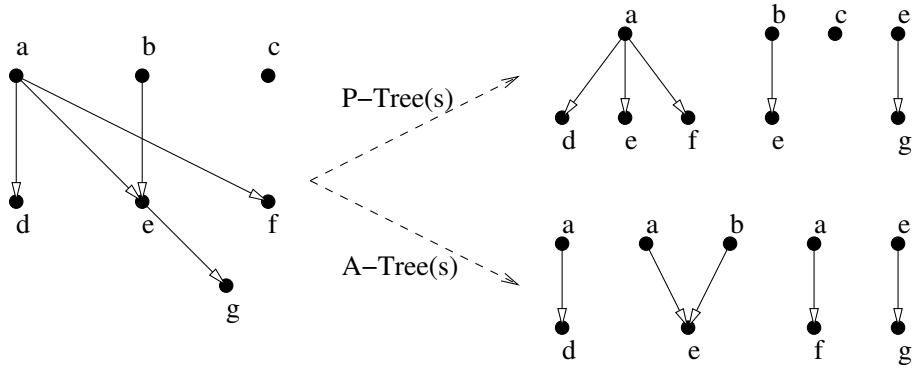


Figure 1: Decomposition of predicate-argument graphs (left) into P-Trees (right top) and A-Trees (right bottom)

$$R(T,H) = \max_{1\le i\le r, 1\le j\le s} \left\{ R(Tree_{T_i}, Tree_{H_j}) \right\}$$

In order to compare two P-Trees or A-Trees, we further define each predicate-argument pair contained in a tree as a semantic dependency triple. Each semantic dependency triple contains a predicate, an argument, and the semantic dependency label in between, in the form of $\langle Predicate, Dependency, Argument \rangle$. Then we define the relatedness function between two trees as the minimum value of the relatedness scores of all the triple pairs from the two trees.

$$R(Tree_T, Tree_H) = \min_{1\le i\le n, 1\le j\le m} \left\{ R(\langle P_T, D_{T_i}, A_{T_i}\rangle, \langle P_H, D_{H_j}, A_{H_j}\rangle) \right\}$$

For the relatedness function between two semantic dependency triples, we define the following two settings: the FULL match and the NOT-FULL match. Either match requires that the predicates are related at the first place. The former means both the dependencies and the arguments are related; while the latter only requires the dependencies to be related.

$$R(\langle P_T, D_T, A_T\rangle, \langle P_H, D_H, A_H\rangle) = \begin{cases} \text{Full} & R(P_T,P_H)=R(D_T,D_H)=R(A_T,A_H)=1 \\ \text{NotFull} & R(P_T,P_H)=R(D_T,D_H)=1 \\ \text{Other} & \text{Otherwise} \end{cases}$$

Now, the only missing components in our definition is the relatedness functions between predicates, arguments, and semantic dependencies. Fortunately, many people have done research on semantic relatedness in lexical semantics that we could use. Therefore, these functions can be realized by different string matching algorithms and/or lexical resources. Since the meaning of relevance is rather wide, apart from the string matching of the lemmas, we also incorporate various resources, from distributionally collected ones to hand-crafted ontologies. We choose VerbOcean (Chklovski and Pantel, 2004) to obtain the relatedness between predicates (after using WordNet (Fellbaum, 1998) to change all the nominal predicates into verbs) and use WordNet for the argument alignment. For the verb relations in VerbOcean, we consider all of them as related; and for WordNet, we not only use the synonyms, hyponyms, and hypernyms, but antonyms as well. Consequently, we simplify these basic relatedness functions into a binary decision. If the corresponding strings are matched or the relations mentioned above exist, the two predicates, arguments, or dependencies are related; otherwise, not.

In addition, the Normalized Google Distance (NGD) (Cilibrasi and Vitanyi, 2007) is applied to both cases[5]. As for the comparison between dependencies, we simply apply the string matching, except for modifier labels, which we treat them as the same[6]. In all, the main idea here is to incorporate both distributional semantics and ontological semantics in order to see whether their contributions are overlapping or complementary. In practice, we use empirical value 0.5 as the threshold. Below the threshold means they are related, oth-

[5]You may find the NGD values of all the content word pairs in RTE-3 and RTE-4 datasets at `http://www.coli.uni-sb.de/~rwang/resources/RTE3_RTE4_NGD.txt`.

[6]This is mainly because it is more difficult for the SRL system to differentiate modifier labels than the complements.

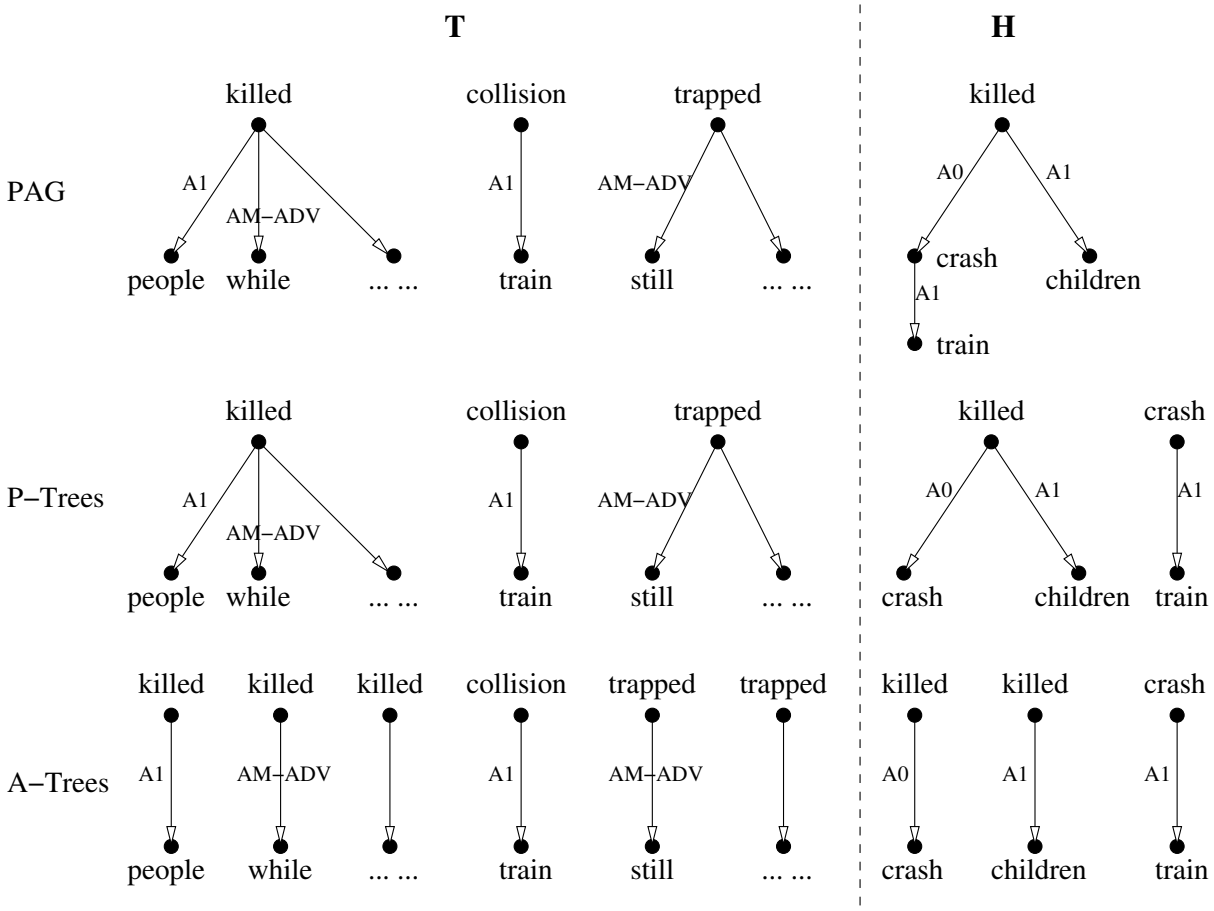


Figure 2: Predicate-argument graphs and corresponding P-Trees and A-trees of the T-H pair

erwise not. In order to achieve a better coverage, we use the OR operator to connect all the relatedness functions above, which means, if any of them holds, the two items are related.

Notice that, although we define only the relatedness between T and H, in principle, the graph representation can also be used for the entailment relationship. However, since it needs more fine-grained analysis and resources, we will leave it as the future work.

5 Experiments

In order to evaluate our method, we setup several experiments. The baseline system here is a simple Naive Bayes classifier with a feature set containing the Bag-of-Words (BoW) overlapping ratio between T and H, and also the syntactic dependency overlapping ratio. The feature model combines two baseline systems proposed by previous work, which gives out quite competitive performance. Since the main goal of this paper is to show the impact of the PAS-based alignment module, we will not compare our results with other RTE systems (In fact, the baseline system already outperforms the average accuracy score of the RTE-4 challenge).

The main data set used for testing here is the RTE-4 data set with three-way annotations (500 entailment T-H pairs (E), 150 contradiction pairs (C), and 350 unknown pairs (U)). The results on RTE-3 data set (combination of the development set and test set, in all, 822 E pairs, 161 C pairs, and 617 U pairs) is also shown, although the original annotation is two-way and the three-way annotation was done by different researchers after the challenge[7].

We will first show the performance of the baseline systems, followed by the results of our PAS-based alignment module and its impact on the whole task. After that, we will also give more detailed analysis of our alignment module, according to different lexical relatedness measurements.

[7]The annotation of the development set was done by students at Stanford, and the annotation of the test set was done as double annotation by NIST assessors, followed by adjudication of disagreements. Answers were kept consistent with the two-way decisions in the main task gold answer file.

5.1 Baselines

The baseline systems used here are based on overlapping ratio of words and syntactic dependencies between T and H. For the word overlapping ratio, we calculate the number of overlapping tokens between T and H and normalize it by dividing it by the number of tokens in H. The syntactic dependency overlapping ratio works similarly: we calculate the number of overlapping syntactic dependencies and divide it by the number of syntactic dependencies in H, i.e. the same as the number of tokens. Enlightened by the relatedness function, we also allow either FULL match (meaning both the dependencies and the parent tokens are matched), and NOTFULL match (meaning only the dependencies are matched). Here we only use string match between lemmas and syntactic dependencies. Table 1 presents the performance of the baseline system.

The results show that, even with the same classifier and the same feature model, with a proper two-stage strategy, it can already achieve better results than the three-way classification. Note that, the first strategy is not so successful, and that is the traditional two-way annotation of the RTE task. Our explanation here is that the BoW method (even with syntactic dependency features) is based on overlapping information shared by T and H, which essentially means the more information they share, the more relevant they are, instead of being more similar or the same. Therefore, for the "$ECU \rightarrow E/CU$" setting, methods based on overlapping information are not the best choice, while for "$ECU \rightarrow U/EC$", they are more appropriate.

In addition, the upper bound numbers show the accuracy when the first-stage classification is perfect, which give us an indication of how far we could go. The lower upper bound for the second strategy is mainly due to the low proportion of the C cases (15%) in the data set; while the other two both show large space for improvement.

5.2 The PAS-based Alignment Module

In this subsection, we present a separate evaluation of our PAS-based alignment module. As we mentioned before (cf. Section 4), there are several parameters to be tuned in our alignment algorithm: a) whether the relatedness function between P-Trees asks for the FULL match; b) whether the function for A-Trees asks for the FULL match; and c) whether both P-Trees and A-Trees being related are required or either of them holds is enough. Since they are all binary values, we use the 3-digit code to represent each setting, e.g. [FFO][8] means *either* P-Trees are FULL matched *or* A-Trees are FULL matched. The performances of different settings of the module are shown in the following Precision-Recall figure 3,

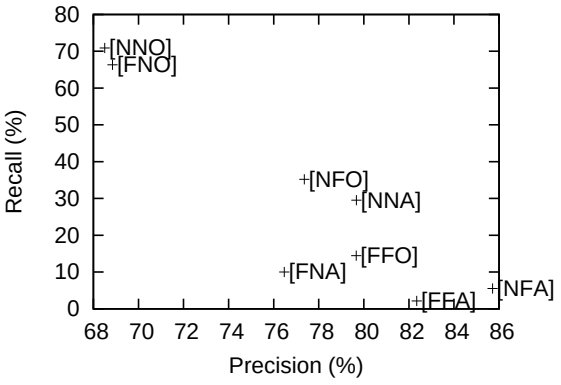


Figure 3: Precision and recall of different alignment settings

Since we will combine this module with the baseline system and it will be integrated as the first-stage classification, the F1 scores are not indicative for selecting the best setting. Intuitively, we may prefer higher precision than recall.

One limitation of our method we need to point out here is that, if some important predicates or arguments in H are not (correctly) identified by the SRL system, fewer P-Trees and A-Trees are required to be related to some part of T, thus, the relatedness of the whole pair could easily be satisfied, leading to false positive cases.

5.3 Impact on the Final Results

The best settings for RTE-3 data set is [NNA] and for RTE-4 data set is [NFO], which are both in the middle of the setting range shown in the previous figure 3.

As for the integration of the PAS-based alignment model with our BoW-based baseline, we only consider the third two-stage classification strategy in Table 1. Other strategies would also be interesting to try, however, the proposed alignment algorithm exploits relatedness between T and H, which might not be fine-grained enough to detect

[8] F stands for FULL, and O stands for OR. Other letters are, N stands for NOTFULL, and A stands for AND.

Strategies	Three-Way	Two-Stage		
	$E/C/U$	$E/CU \rightarrow E/C/U$	$C/EU \rightarrow C/E/U$	$U/EC \rightarrow U/E/C$
Accuray	53.20%	50.00%	53.50%	54.20%
Upper Bound	/	82.80%	68.70%	84.90%

Table 1: Performances of the Baselines

entailment or contradiction. New alignment algorithm has to be designed to explore other strategies. Thus, in this work, we believe that the alignment algorithm based on PAS (and other methods based on overlapping information between T and H) is suitable for the $U/EC \rightarrow U/E/C$ classification strategy.

Table 2 shows the final results.

The first observation is that the improvement of accuracy on the first stage of the classification can be preserved to the final results. And our PAS-based alignment module can help, though there is still large space for improvement. Compared with the significantly improved results on RTE-4, the improvement on RTE-3 is less obvious, mainly due to the relatively lower precision (70.33% vs. 79.67%) of the alignment module itself.

Also, we have to say that the improvement is not as big as we expected. There are several reasons for this. Besides the limitation of our approach mentioned in the previous section, the predicates and arguments themselves might be too sparse to convey all the information we need for the entailment detection. In addition, in some sense, the baseline is quite strong for this comparison, since the PAS-based alignment module relies on the overlapping words at the first place, there are quite a few pairs solved by both the main approach and the baseline. Then, it would be interesting to take a closer look at the lexical resources used in the main system, which is another additional knowledge it has, comparing with the baseline.

5.4 Impact of the Lexical Resources

We did an ablation test of the lexical resources used in our alignment module. Recall that we have applied three lexical resources, VerbOcean for the predicate relatedness function, WordNet for the argument relatedness function, and Normalized Google Distance for both. Table 3 shows the performances of the system without each of the resources,

The results clearly show that each lexical resource does contribute some improvement to the final performance of the system and it confirms the idea of combining lexical resources being acquired in different ways. For instance, at the beginning, we expected that the relationship between "*people*" and "*children*" could be captured by WordNet, but in fact not. Fortunately, the NGD has a quite low value of this pair of words (0.21), which suggests that they occur together quite often, or in other words, they are relevant.

One interesting future work on this aspect is to substitute the OR connector between these lexical resources with an AND operator. Thus, instead of using them to achieve a higher coverage, whether they could be filters for each other to increase the precision will also be interesting to know.

6 Conclusion and Future Work

In this paper, we address the motivation and issues of casting the three-way RTE problem into a two-stage binary classification task. We apply an SRL system to derive the predicate-argument structure of the input sentences, and propose ways of calculating semantic relatedness between the shallow semantic structures of T and H. The experiments show improvements in the first-stage classification, which accordingly contribute to the final results of the RTE task.

For future work, we would like to see whether the PAS can help the second-stage classification as well, e.g. the semantic dependency of negation (AM-NEG) could be helpful for the contraction recognition. Furthermore, since the PAS is usually a bag of unconnected graphs, we could find a way to joint them together, in order to consider both inter- and intra- sentential inferences based on it.

In addition, this approach has the potential to be integrated with other RTE modules. For instance, for the predicate alignment, we may consider to use DIRT rules (Lin and Pantel, 2001) or other paraphrase resources (Callison-Burch, 2008), and for the argument alignment, external named-entity recognizer and anaphora resolver would be very helpful. Even more, we also plan to compare/combine it with other methods which are not based on overlapping information between T and H.

Systems	Baseline1	Baseline2	SRL+Baseline2	The First Stage		
Data Sets	Three-Way	Two-Stage	Two-Stage	Baseline2	SRL+Baseline2	SRL
RTE-3 [NNA]	52.19%	52.50%	53.69%(2.87%↑)	59.50%	60.56%(1.78%↑)	70.33%
RTE-4 [NFO]	53.20%	54.20%	56.60%(6.39%↑)	67.10%	70.20%(4.62%↑)	79.67%

Table 2: Results on the Whole Datasets

Data Sets	SRL+Baseline	SRL+Baseline - VO	SRL+Baseline - NGD	SRL+Baseline - WN
RTE-3 [NNA]	53.69%	53.19%(0.93%↓)	53.50%(0.35%↓)	52.88%(1.51%↓)
RTE-4 [NFO]	56.60%	56.00%(1.06%↓)	56.10%(0.88%↓)	55.70%(1.59%↓)

Table 3: Impact of the Lexical Resources

Acknowledgments

The first author is funded by the PIRE PhD scholarship program sponsored by the German Research Foundation (DFG). The second author thanks the German Excellence Cluster of Multimodal Computing and Interaction for the support of the work.

References

Eugene Agichtein, Walt Askew, and Yandong Liu. 2009. Combining Lexical, Syntactic, and Semantic Evidence for Textual Entailment Classification. In *Proceedings of the First Text Analysis Conference (TAC 2008)*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montreal, Canada.

Aljoscha Burchardt, Nils Reiter, Stefan Thater, and Anette Frank. 2007. A semantic approach to textual entailment: System evaluation and task analysis. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Prague, Czech Republic.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of EMNLP*.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, Barcelona, Spain.

Rudi Cilibrasi and Paul M. B. Vitanyi. 2007. The Google Similarity Distance. *IEEE/ACM Transactions on Knowledge and Data Engineering*, 19(3):370–383.

Peter Clark and Phil Harrison. 2009. Recognizing Textual Entailment with Logical Inference. In *Proceedings of the First Text Analysis Conference (TAC 2008)*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. In *Machine Learning Challenges*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer.

Marie-Catherine de Marneffe, Anna N. Rafferty, and Christopher D. Manning. 2008. Finding contradictions in text. In *Proceedings of ACL-08*.

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

Danilo Giampiccolo, Hoa Trang Dang, Bernardog Magnini, Ido Dagan, Elena Cabrio, and Bill Dolan. 2009. The Fourth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the First Text Analysis Conference (TAC 2008)*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning*, Boulder, CO, USA.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for Using Textual Entailment in Open-Domain Question Answering. In *Proceedings of COLING-ACL 2006*, pages 905–912, Sydney, Australia.

Dekang Lin and Patrick Pantel. 2001. DIRT - Discovery of Inference Rules from Text. In *In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Bill MacCartney, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP 2008*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of hlt-emnlp 2005*, pages 523–530, Vancouver, Canada.

Sebastian Pado, Marie-Catherine de Marneffe, Bill MacCartney, Anna N. Rafferty, Eric Yeh, and

Christopher D. Manning. 2009. Deciding entailment and contradiction with stochastic and edit distance-based alignment. In *Proceedings of the First Text Analysis Conference (TAC 2008)*.

Reda Siblini and Leila Kosseim. 2009. Using Ontology Alignment for the TAC RTE Challenge. In *Proceedings of the First Text Analysis Conference (TAC 2008)*.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th conference on computational natural language learning (CoNLL-2008)*, Manchester, UK.

Rui Wang and Günter Neumann. 2007. Recognizing textual entailment using a subsequence kernel method. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-07)*, pages 937–942, Vancouver, Canada.

Yi Zhang, Rui Wang, and Hans Uszkoreit. 2008. Hybrid Learning of Dependency Structures from Heterogeneous Linguistic Resources. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 2008)*, pages 198–202, Manchester, UK.

Learning Term-weighting Functions for Similarity Measures

Wen-tau Yih
Microsoft Research
Redmond, WA, USA
scottyih@microsoft.com

Abstract

Measuring the similarity between two texts is a fundamental problem in many NLP and IR applications. Among the existing approaches, the cosine measure of the term vectors representing the original texts has been widely used, where the score of each term is often determined by a TFIDF formula. Despite its simplicity, the quality of such cosine similarity measure is usually domain dependent and decided by the choice of the term-weighting function. In this paper, we propose a novel framework that learns the term-weighting function. Given the labeled pairs of texts as training data, the learning procedure tunes the model parameters by minimizing the specified loss function of the similarity score. Compared to traditional TFIDF term-weighting schemes, our approach shows a significant improvement on tasks such as judging the quality of query suggestions and filtering irrelevant ads for online advertising.

1 Introduction

Measuring the semantic similarity between two texts is an important problem that has many useful applications in both NLP and IR communities. For example, Lin (1998) defined a similarity measure for automatic thesaurus creation from a corpus. Mihalcea et al. (2006) developed several corpus-based and knowledge-based word similarity measures and applied them to a paraphrase recognition task. In the domain of web search, different methods of measuring similarity between short text segments have recently been proposed for solving problems like query suggestion and alternation (Jones et al., 2006; Sahami and Heilman, 2006; Metzler et al., 2007; Yih and Meek, 2007).

Among these similarity measures proposed in various applications, the vector-based methods are arguably the most widely used. In this approach, the text being compared with is first represented by a term vector, where each term is associated with a weight that indicates its importance. The similarity function could be cosine (i.e., the inner product of two normalized unit term vectors, or equivalently a linear kernel), or other kernel functions such as the Gaussian kernel.

There are essentially two main factors that decide the quality of a vector-based similarity measure. One is the *vector operation* that takes as input the term vectors and computes the final similarity score (e.g., cosine). The other is how these term vectors are constructed, including the term selection process and how the weights are determined. For instance, a TFIDF scheme for measuring document similarity may follow the bag-of-words strategy to include all the words in the document when constructing the term vectors. The weight of each term is simply the product of its term frequency (i.e., the number of occurrences in the document) and inverse document frequency (i.e., the number of documents in a collection that contain this term).

Despite its simplicity and reasonable performance, such approach suffers from several weaknesses. For instance, the similarity measure is not domain-dependent and cannot be easily adjusted to better fit the final objective, such as being a metric value used for clustering or providing better ranking results. Researchers often need to experiment with variants of TFIDF formulas and different term selection strategies (e.g., removing stopwords or stemming) to achieve acceptable performance (Manning et al., 2008). In addition, when more information is available, such as the position of a term in the document or whether a term is part of an anchor text, incorporating it in the similarity measure in a principled manner may not be easy.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 793–802,
Singapore, 6-7 August 2009.

In this paper, we propose a general ***t***erm-***w***eighting l***ea***rning framewor***k***, TWEAK, that learns the term-weighting function for the vector-based similarity measures. Instead of using a fixed formula to decide the weight of each term, TWEAK uses a parametric function of features of each term, where the model parameters are learned from labeled data. Although the weight of each term conceptually represents its importance with respect to the document, tuning the model parameters to optimize for such objectives may not be the best strategy due to two reasons. While the label of whether a pair of texts is similar is not difficult to collect from human annotators[1], the label of whether a term in a document is important is often very ambiguous and hard to decide. Even if such annotation issue can be resolved, aligning the term weights with the *true* importance of each term may not necessarily lead to our real objective – deriving a better similarity measure for the target application. Therefore, our learning framework, TWEAK, assumes that we are given only the labels of the pairs of texts being compared, such as whether the two texts are considered similar by human subjects.

TWEAK is flexible in choosing various loss functions that are close to the true objectives, while still maintaining the simplicity of the vector-based similarity measures. For example, a system that implements the TFIDF cosine measure can easily replace the original term-weighting scores with the ones output by TWEAK without changing other portions of the algorithm. TWEAK is also novel compared to other existing learning methods for similarity measures. For instance, we do not learn the scores of all the terms in the vocabulary directly, which is one of the methods proposed by Bilenko and Mooney (2003). Because the vocabulary size is typically large in the text domain (e.g., all possible words in English), learning directly the term-weighting scores may suffer from the data sparsity issue and cannot generalize well in practice. Instead, we focus on learning the model parameters for features that each term may have, which results in a much smaller feature space. TWEAK also differs from the model combination approach proposed by Yih and Meek (2007), where the output scores of different similarity measures are combined via a learned linear function. In contrast, TWEAK effectively learns a new similarity measure by tuning the term-weighting function and can potentially be complementary to the model combination approach.

As will be demonstrated in our experiments, in applications such as judging the relevance of different query suggestions and determining whether a paid-search ad is related to the user query, TWEAK can incorporate various kinds of term–document information and learn a term-weighting function that significantly outperforms the traditional TFIDF scheme in several evaluation metrics, when using the same vector operation (i.e., cosine) and the same set of terms.

We organize the rest of the paper as follows. Sec. 2 first gives a high-level view of our term-weighting learning framework. We then formally define our model and present the loss functions that can be optimized for in Sec. 3. Experiments on target applications are presented in Sec. 4. Finally, we compare our approach with some related work in Sec. 5 and conclude the paper in Sec. 6.

2 Problem Statement

To simplify the description, assume that the texts we are comparing are two documents. A general architecture of vector-based similarity measures can be formally described as follows. Given two documents D_p and D_q, a similarity function maps them to a real-valued number, where a higher value indicates these two documents are semantically more related, considered by the measure.

Suppose a pre-defined vocabulary set $\mathcal{V} = \{t_1, t_2, \cdots, t_n\}$ consists of all possible terms (e.g., tokens, words) that may occur in the documents. Each document D_p is represented by a term vector of length n: $\mathbf{v}_p = (s_p^1, s_p^2, \cdots, s_p^n)$, where $s_p^i \in \mathbf{R}$ is the weight of term t_i, and is determined by the term-weighting function tw that depends on the term and the document (i.e., $s_p^i \equiv tw(t_i, D_p)$). The similarity between documents D_p and D_q is then computed by a vector operation function $f_{sim} : (\mathbf{v}_p, \mathbf{v}_q) \rightarrow \mathbf{R}$, illustrated in Fig. 1.

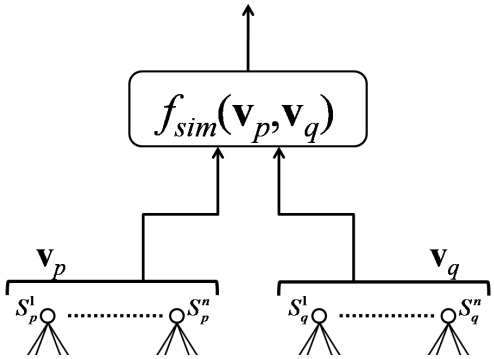
Determining the specific functions f_{sim} and tw effectively decides the final similarity measure. For example, the functions that construct the traditional TFIDF cosine similarity can be:

$$f_{sim}(\mathbf{v}_p, \mathbf{v}_q) \equiv \frac{\mathbf{v}_p \cdot \mathbf{v}_q}{||\mathbf{v}_p|| \cdot ||\mathbf{v}_q||} \quad (1)$$

$$tw(t_i, D_p) \equiv tf(t_i, D_p) \cdot \log\left(\frac{N}{df(t_i)}\right) \quad (2)$$

[1] As argued in (Sheng et al., 2008), low-cost labels may nowadays be provided by outsourcing systems such as Amazon's Mechanical Turk or online ESP games.

Figure 1: A general architecture of vector-based similarity measures



where N is the size of the document collection for deriving document frequencies, tf and df are the functions computing the term frequency and document frequency, respectively.

In contrast, TWEAK also takes a specified vector function f_{sim} but assumes a parametric term-weighting function $tw_{\mathbf{w}}$. Given the training data, it learns the model parameters $\mathbf{w}$ that optimize for the designated loss function.

3 Model

As a specific instantiation of our learning framework, the term-weighting function used in this paper is a linear combination of features extracted from the input term and document. In particular, the weight of term t_i with respect to document D_p is

$$s_p^i = tw_{\mathbf{w}}(t_i, D_p) \equiv \sum_j w_j \phi_j(t_i, D_p), \qquad (3)$$

where ϕ_j is the j-th feature function and w_j is the corresponding model parameter.

As for the vector operation function f_{sim}, we use the same cosine function (Eq. 1). Notice that we choose these functional forms for their simplicity and good empirical performance shown in preliminary experiments. However, other smooth functions can certainly be used.

The choice of loss function for training model parameters depends on the true objective in the target application. In this work, we consider two different learning settings: learning directly the *similarity metric* and learning the *preference ordering*, and compare several loss functions experimentally.

3.1 Learning Similarity Metric

In this setting, we assume that the learning algorithm is given a set of document pairs. Each of them is associated with a label that indicates whether these two documents are similar (e.g., a binary label where 1 means similar and 0 otherwise) or the degree of similarity (e.g., a real-valued label ranges from 0 to 1), considered by the human subjects. A training set of m examples can be denoted as $\{(y_1, (D_{p_1}, D_{q_1})), (y_2, (D_{p_2}, D_{q_2})), \cdots, (y_m, (D_{p_m}, D_{q_m}))\}$, where y_k is the label and (D_{p_k}, D_{q_k}) is the pair of documents to compare. Following the vector construction described in Eq. 3, let $\mathbf{v}_{p_1}, \mathbf{v}_{q_1}, \cdots, \mathbf{v}_{p_m}, \mathbf{v}_{q_m}$ be the corresponding term vectors of these documents.

We consider two commonly used loss functions, *sum-of-squares error* and *log loss*[2]:

$$L_{sse}(\mathbf{w}) = \frac{1}{2}\sum_k^m (y_k - f_{sim}(\mathbf{v}_{p_k}, \mathbf{v}_{q_k}))^2 \qquad (4)$$

$$L_{log}(\mathbf{w}) = \sum_k^m -y_k \log(f_{sim}(\mathbf{v}_{p_k}, \mathbf{v}_{q_k})) - (1 - y_k)\log(1 - f_{sim}(\mathbf{v}_{p_k}, \mathbf{v}_{q_k})) \qquad (5)$$

Eq. 4 and Eq. 5 can further be regularized by adding $\frac{\alpha}{2}||\mathbf{w}||^2$ in the loss function, which may improve the performance empirically and also constrain the range of the final term-weighting scores. Learning the model parameters for minimizing these loss functions can be done using standard gradient-based optimization methods. We choose the L-BFGS (Nocedal and Wright, 2006) method in our experiments for its guarantee to find a local minimum and fast convergence. The derivation of gradients is fairly straightforward, which we skip here.

Notice that other loss functions can also be used in this framework. Interested readers can refer to, say, (Bishop, 1995), for other loss functions and their theoretical justifications.

3.2 Learning Preference Ordering

In many applications where the similarity measure is applied, the goal is to obtain a *ranked* list of the candidate elements. For example, in the task of

[2] Although in theory the cosine function may return a negative value and make the log-loss uncomputable, this can be easily avoided in practice by selecting appropriate initial model parameters and by constraining the term-weighting scores to be non-negative.

filtering irrelevant ads, a good similarity measure is expected to rank appropriate ads higher than the irrelevant ones. A desired trade-off of false-positive (mistakenly filtered good ads) and false-negative (unfiltered bad ads) can be achieved by selecting a decision threshold. The exact value of the similarity measure, in this case, is not crucial. For these applications, it is more important if the model parameters can better predict the *pair-wise preference*. Learning preference ordering is also motivated by the observation that preference annotations are generally more reliable than categorical similarity labels (Carterette et al., 2008) and has been advocated recently by researchers (e.g., Burges et al. (2005)).

In the setting of learning preference ordering, we assume that each training example consists of *two* pairs of documents, associated with a label indicating which pair of documents is considered more preferable. A training set of m examples can be formally denoted as $\{(y_1, (x_{a_1}, x_{b_1})), (y_2, (x_{a_2}, x_{b_2})), \cdots, (y_m, (x_{a_m}, x_{b_m}))\}$, where $x_{a_k} = (D_{p_{a_k}}, D_{q_{a_k}})$ and $x_{b_k} = (D_{p_{b_k}}, D_{q_{b_k}})$ are two pairs of documents and $y_k \in \{0, 1\}$ indicates the pairwise order preference, where 1 means x_{a_k} should be ranked higher than x_{b_k} and 0 otherwise.

We use a loss function that is very similar to the one proposed by Dekel et al. (2004) for label ranking. Let Δ_k be the difference of the similarity scores of these two document pairs. Namely,

$$\Delta_k = f_{sim}(\mathbf{v}_{p_{a_k}}, \mathbf{v}_{q_{a_k}}) - f_{sim}(\mathbf{v}_{p_{b_k}}, \mathbf{v}_{q_{b_k}})$$

The loss function L, which can be shown to upper bound the pairwise accuracy (i.e., the 0-1 loss of the pairwise predictions), is:

$$L(\mathbf{w}) = \sum_{k=1}^{m} \log(1+\exp(-y_k \cdot \Delta_k - (1-y_k) \cdot (-\Delta_k))) \quad (6)$$

Similarly, Eq. 6 can be regularized by adding $\frac{\alpha}{2}||\mathbf{w}||^2$ in the loss function.

4 Experiments

We demonstrate how to apply our term-weighting learning framework, TWEAK, to measuring similarity for short text segments and to judging the relevance of an ad landing page given an query. In addition, we compare experimentally the performance of using different training settings, loss functions and features against the traditional TFIDF term-weighting scheme.

4.1 Similarity for Short Text Segments

Judging the similarity between two short text segments is a crucial problem for many search and online advertising applications. For instance, *query reformulation* or *query substitution* needs to measure the similarity between two queries. A product keyword recommendation system needs to determine whether the given product name and the suggested keyword is related.

Because the length of the text segment is typically short, ranging from a single word to a dozen words, naively applying methods based on word overlapping such as the Jaccard coefficient leads to poor results (Sahami and Heilman, 2006; Yih and Meek, 2007). To overcome this difficulty, Sahami and Heilman (2006) proposes a Web-kernel function, which first expands the short text segment by issuing it to a search engine as the query, and then collectes the snippets of the top results to construct a pseudo-document. TFIDF term vectors of the pseudo-documents are used to represent the original short text segments and the cosine score of these two vectors is used as the similarity measure.

In this section, we apply TWEAK to this problem by replacing the TFIDF term-weighting scheme with the learned term-weighting function, when constructing the vectors from the pseudo-documents. Our target application is *query suggestion* – automatically presenting queries that are related to the one issued by the user. In particular, we would like to use our similarity measure as a filter to determine whether queries suggested by various algorithms and heuristics are indeed closely related to the target query.

4.1.1 Task & Data

Our query suggestion dataset has been previously used in (Metzler et al., 2007; Yih and Meek, 2007) and is collected in the following way. From the search logs of a commercial search engine, a random sample of 363 thousand queries from the top 1 million most frequent queries in late 2005 were first taken as the query and suggestion candidates. Among them, 122 queries were chosen randomly as our target queries; each of them had up to 100 queries used as suggestions, generated by various query suggestion mechanisms.

Given these pairs of query and suggestions, human annotators judged the level of similarity using a 4-point scale – *Excellent*, *Good*, *Fair* and *Bad*,

where Excellent and Good suggestions are considered clearly related to the query intent, while the other two categories mean the suggestions are either too general or totally unrelated. In the end, 4,852 query/suggestion pairs that had effective annotations were collected. The distribution of the four labels is: Excellent - 5%, Good - 12%, Fair - 44% and Bad - 39%.

For the simplicity of both presentation and implementation, query/suggestion pairs labeled as Excellent or Good are treated as positive examples and the rest as negative ones. Notice that TWEAK is not restricted in using only binary labels. For instance, the pairwise preference learning setting only needs to know which pair of objects being compared is more preferred. The model and algorithm do not have to change regardless of whether the label reflects the degree of similarity (e.g, the original 4-scale labels) or binary categories. For the metric learning setting, an ordinal regression approach (e.g, (Herbrich et al., 2000)) can be applied for multi-category labels.

We used the same query expansion method as described in (Sahami and Heilman, 2006). Each query/suggestion was first issued to a commercial search engine. The result page with up to 200 snippets (i.e., titles and summaries) was used as the pseudo-document to create the term vector that represents the original query/suggestion. As described earlier in Eq. 3, the weight of each term is a linear function of a set of predefined features, which are described next.

4.1.2 Features

Because the pseudo-documents are constructed using the search result snippets instead of regular web documents, special formatting or link information provided by HTML is not very meaningful. Therefore, we focused on using features that are available for plain-text documents, including:

- **Bias**: 1 for all examples.
- **TF**: We used $\log(tf + 1)$ as the *term frequency* feature, where tf is the number of times the term occurs in the original pseudo-document.
- **DF**: We used $\log(df + 1)$ as the *document frequency* feature, where df is the number of documents in our collection that contain this term.
- **QF**: The search engine query log reflects the distribution of the words/phrases in which people are interested (Goodman and Carvalho, 2005; Yih et al., 2006). We took a log file with the most frequent 7.5 million queries and used $\log(qf + 1)$ as feature, where qf is the query frequency.
- **Cap**: A capitalized word may indicate being part of a proper noun or being more important. When the term is capitalized in at least one occurrence in the pseudo-document, the value of this feature is 1; otherwise, it is 0.
- **Loc & Len**: The beginning of a regular document often contains a summary with important words. In the pseudo-documents created using search snippets, words that occur in the beginning come from the top results, which are potentially more relevant to the original query/suggestion. We created two specific features using this location information. Let loc be the word position of the target term and len be the total number of words of this pseudo-document. The logarithmic value $\log(loc + 1)$ and the ratio loc/len were both used as features. In order for the learning procedure to adjust the scaling, the logarithmic value of the document length, $\log(len + 1)$, was also used.

4.1.3 Results

We conducted the experiments using 10-fold cross-validation. The whole query/suggestion pairs were first split into 10 subsets of roughly equal sizes. Pairs with the same target query were put in the same subset. In each round, one subset was used for testing. 95% of the remaining data was used for training the model and 5% was used as the development set. We trained six models with different values of the regularization hyperparameter $\alpha \in \{0.003, 0.01, 0.03, 0.1, 0.3, 1\}$ and determined which model to use based on its performance on the development set, although the result actually did not vary a lot as α changed.

We compared three learning configurations – metric learning with sum-of-squares error (Metric$_{sse}$) and log loss (Metric$_{log}$) and the pairwise preference learning (Preference). The learned term-weighting functions were used to compare with the Web-kernel similarity function, which implemented the TFIDF term-weighting scheme using Eq. 2.

Table 1: The AUC scores, mean averaged precision and precision at 3 of similarity measures using different term-weighting functions. The numbers with the † sign are statistically significantly better compared to the Web-kernel method.

Method	AUC	MAP	Prec@3
Web-kernel	0.732	0.540	0.556
Metric$_{sse}$	0.775†	0.590	0.553
Metric$_{log}$	0.781†	0.585	0.545
Preference	**0.782**†	**0.597**†	**0.570**

We evaluated these models using three different evaluation metrics: the AUC score, precision at k and MAP (mean averaged precision). The area under the ROC curve (AUC) is typically used to judge the overall quality of a ranking function. It has been shown equivalent to the averaged accuracy of the pairwise preference predictions of all possible element pairs in the sequence, and can be calculated by the the following Wilcoxon-Mann-Whitney statistic (Cortes and Mohri, 2004):

$$A(f;\mathbf{x},\mathbf{y}) = \sum_{i,j:y_i>y_j} \mathbf{I}_{f(x_i)>f(x_j)} + \frac{1}{2}\mathbf{I}_{f(x_i)=f(x_j)},$$

where f is the similarity measure, $\mathbf{x}$ is the sequence of compared elements and $\mathbf{y}$ is the labels.

Another metric that is commonly used in a ranking scenario is *precision at* k, which computes the accuracy of the top-ranked k elements and ignores the rest. We used $k = 3$ in our task, which means that for each target query, we selected three suggestions with the highest similarity scores and computed the averaged accuracy.

One issue of precision at k is that it does not provide an overall quality measure of the ranking function. Therefore, we also present MAP (mean averaged precision), which is a single number that summarizes the performance of the ranking function by considering both precision and recall, and has been shown reliable in evaluating various information retrieval tasks (Manning et al., 2008). Suppose there are m relevant elements in a sequence, where $r_1, r_2, \cdots, r_m$ are their locations. The averaged precision is then:

$$AP = \frac{1}{m}\sum_{j=1}^{m} \mathrm{Prec}(r_j),$$

where $\mathrm{Prec}(r_j)$ is the precision at r_j. We computed the averaged precision values of the 10 test sets in our cross-validation setting and report their mean value.

As shown in Table 1, all three learned term-weighting functions lead to better similarity measures compared to the TFIDF scheme in terms of the AUC and MAP scores, where the preference order learning setting performs the best. However, for the precision at 3 metric, only the preference learning setting has a higher score than the TFIDF scheme, but the difference is not statistically significant[3]. This is somewhat understandable since the design of our loss function focuses on the overall quality instead of only the performance of the top ranked elements.

4.2 Query/Page Similarity

Measuring whether a page is relevant to a given query is the main problem in information retrieval and has been studied extensively. Instead of retrieving web pages that are relevant to the query according to the similarity measure, our goal is to implement a paid-search ad filter for commercial search engines. In this scenario, textual ads with bid keywords that match the query can enter the auction and have a chance to be shown on the search result page. However, as the advertisers may bid on keywords that are not related to their advertisements, it is important for the system to filter irrelevant ads to ensure that users only receive useful information. For this purpose, we measure the similarity between the query and the ad landing page (i.e., the page pointed by the ad) and remove the ad when the score of its landing page is below a pre-selected threshold[4].

Given a pair of query and ad landing page, while the *query* term vector is constructed using the same query expansion technique described in Sec. 4.1, the *page* term vector can be created directly from the web page since it is a regular document that contains enough content. As usual, our goal is to produce a better similarity measure by learning the term-weighting functions for these two types of vectors jointly.

[3]We conducted a paired-t test on the 10 individual scores from the cross-validation results of each learned term-weighting function versus the Web-kernel method. The results are considered statistically significant when the p-value is lower than 0.05.

[4]One may argue that the filter should measure the similarity between the query and ad-text. However, an ad will not provide useful information to the user if the final destination page is not relevant to the query, even if its ad-text looks appealing.

4.2.1 Data

We first collected a random sample of queries and paid-search ads shown on a commercial search engine during 2008, as well as the ad landing pages. Judged by several human annotators, each page was labeled as relevant or not compared to the issued query. After removing some pairs where the query intent was not clear or the landing page was no longer available, we managed to collect 13,341 query/page pairs with reliable labels. Among them, 8,309 were considered relevant and 5,032 were labeled irrelevant.

4.2.2 Features

In this experiment, we tested the effect of using different features and experimented with three feature sets: *TF&DF*, *Plain-text* and *HTML*. *TF&DF* contains only $\log(tf+1)$, $\log(df+1)$ and the bias feature. The goal of using this feature set is to test whether we can learn a better term-weighting function given the *same* amount of information as the TFIDF scheme has. The second feature set, *Plain-text*, consists of all the features described in Sec. 4.1.2. As mentioned earlier, this set of features can be used for regular text documents that do not have special formatting information. Finally, feature set *HTML* is composed of all the features used in *Plain-text* plus features extracted from some special properties of web documents, including:

- **Hypertext**: The anchor text in an HTML document usually provides important information. If there is at least one occurrence of the term that appears in some anchor text, the value of this feature is 1; otherwise, it is 0.
- **URL**: A web document has a uniquely useful property – the name of the document, which is its URL. If the term is a substring of the URL, then the value of this feature is 1; otherwise, it is 0.
- **Title**: The value of this feature is 1 when the term is part of the title; otherwise, it is 0.
- **Meta**: Besides Title, several meta tags used in the HTML header explicitly show the important words selected by the page author. Specifically, whether the term is part of a meta-keyword is used as a binary feature. Whether the term is in the meta-description segment is also used.

Table 2: The AUC scores, true-positive rates at false-positive rates 0.1 and 0.2 of the ad filter based on different term-weighting functions. The difference between any pair of numbers of the same evaluation metric is statistically significant.

Method	AUC	$\text{TPR}_{\text{fnr}=0.1}$	$\text{TPR}_{\text{fnr}=0.2}$
TFIDF	0.794	0.527	0.658
TF&DF	0.806	0.430	0.639
Plain-text	0.832	0.503	0.704
HTML	**0.855**	**0.568**	**0.750**

Because the term vector that represents the query is created from the pseudo-document (i.e., a collection of search snippets), the values of these HTML-specific features are all 0 for the query term vector. This set of features are only useful for deciding the weights of the terms in a page term vector.

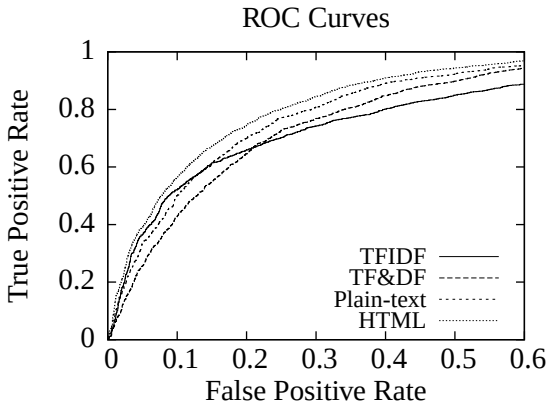
4.2.3 Results

We split our data into 10 subsets and conducted the experiments using the same 10-fold cross-validation setting described in Sec. 4.1.3, including how we used the development set to select the regularization hyper-parameter α. The pairs that have the same target query were again put in the same subsets. We used only the preference ordering learning setting for its good performance shown in the previous set of experiments. Models compared here were learned from the three different sets of features, as well as the same fixed TFIDF term-weighting formula (i.e., Eq. 2) used in Sec. 4.1. Table 2 reports the averaged results of the 10 testing sets in AUC, as well as the true-positive rates at two low false-positive rate points (FPR=0.1 and FPR=0.2). The difference between any pair of numbers of the same evaluation metric is statistically significant[5].

As we can see from the table, having more features does lead to a better term-weighting function. With all features (i.e., *HTML*), the model achieves the highest AUC score among all configurations. Features available in plain-text documents (i.e., *Plain-text*) other than term frequency and document frequency can still improve the performance significantly. When only the TF and DF features are available, the learned term-weighting function still outperforms the TFIDF scheme, al-

[5]We conduct paired-t tests as described in Sec. 4.1.3. All the p-values after Bonferroni correction are less than 0.01.

Figure 2: ROC Curves of the ad filters using different term-weighting functions



though the improvement gain is much smaller compared to the other two settings.

Notice that the behaviors of these models at different false-positive regions varies from the traditional TFIDF scheme. At a low false-positive point (e.g., FPR=10%), only the model that uses all features performs better than TFIDF. This phenomenon can be clearly observed from the ROC curves plotted in Fig. 2, where the models were trained using half of the data and applied to the other half to generate the similarity scores. If only the performance at a very low false-positive rate matters, TWEAK can still be easily adjusted by modifying the loss function using techniques such as training with utility (Domingos, 1999; Morik et al., 1999).

5 Related Work

Our term-weighting learning framework can be analogous to the "Siamese" architecture for learning jointly two neural networks that share the same set of model weights (Bromley et al., 1993). For instance, a term vector can be viewed as a very large single-layer neural network, where each term in the vocabulary is a node that takes as input the features and outputs the learned term-weighting score. Previous applications of this learning machine are typically problems in image processing or computer vision. For example, Chopra et al. (2005) designed an algorithm to learn a similarity metric for face verification, which is based on the difference between two vectors. In our earlier experiments (not reported in this paper) of using vector difference instead of cosine, we did not observe positive outcomes. We hypothesize that because the length of the term vector in our problem can be extremely large (i.e., the size of the vocabulary), a similarity measure based on vector difference can easily be affected by terms that do not occur in both documents, even when the co-occurred terms have very large weights.

Learning similarity measures for text has also been proposed by several researchers. For instance, Bilenko and Mooney (2003) applied SVMs to directly learn the weights of co-occurred words in two text records, which are then used for measuring similarity for duplicate detection. Although this approach worked moderately well in the database domain, it may not be suitable to handle general text similarity problems for two reasons. First, the vocabulary size is typically large, which results in a very high dimensional feature space for the learning problem. It is very likely that some rarely used and yet important terms occur in the testing documents but not in the training data. The weights of those terms may not be reliable or even be learned. Second, this learning approach can only learn the importance of the terms from the labels of whether two texts are considered similar, how to incorporate the basic information of these terms such as the position or query log frequency is not clear.

An alternative learning approach is to combine multiple similarity measures with learned coefficients (Yih and Meek, 2007), or to apply the technique of *kernel alignment* (Cristianini et al., 2002) to combining a set of kernel functions for tuning a more appropriate kernel based on labeled data. This type of approaches can be viewed as constructing an ensemble of different existing similarity measures without modifying the term weighting function, and may not generate mathematically equivalent similarity functions as derived by TWEAK. Although learning in this approach is usually very fast due to the model form and the small number of parameters to learn, its improvement is limited by the quality of the individual similarity measures. In spite of the fundamental difference between our approach and this combination method, it is worth noticing that these two approaches are in fact complementary to each other. Having a newly learned term-weighting function effectively provides a new similarity measure and therefore can be combined with other measures.

6 Conclusions

In this paper, we presented a novel term-weighting learning framework, TWEAK, for improving similarity measures based on term vectors. Given the labels of text pairs for training, our method learns the model parameters to calculate the score of each term, optimizing the desired loss function that is suitable for the target application. As we demonstrated in the experiments, TWEAK with different features and training settings significantly outperforms the traditional TFIDF term-weighting scheme.

TWEAK also enjoys several advantages compared to existing methods. From an engineering perspective, adopting the new term-weighting scores produced by our model is straightforward. If a similarity measure has been implemented, the algorithm need not be changed – only the term vectors need to be updated. From the learning perspective, additional information regarding each term with respect to the document can now be incorporated easily via feature functions. Weights (i.e., model parameters) of these features are learned in a principled way instead of being adjusted manually. Finally, TWEAK is potentially complementary to other methods for improving the similarity measure, such as model combination of various types of similarity measures (Yih and Meek, 2007) or different term vector construction methods such as Latent Semantic Analysis (Deerwester et al., 1990).

In the future, we plan to explore more vector operations other than the inner-product (i.e., cosine) as well as different functional forms of the term-weighting function (e.g. log-linear instead of linear). Designing new loss functions to better fit the true objectives in various target applications and studying the quality of a similarity measure based on both term-weighting learning and model combination are also on our agenda. In terms of applications, we would like to apply TWEAK in other problems such as paraphrase recognition and near-duplicate detection.

Acknowledgments

The author thanks the anonymous reviewers for their valuable comments and is grateful to Asela Gunawardana, Chris Meek, John Platt and Misha Bilenko for many useful discussions.

References

Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of KDD-2003*, pages 39–48.

Christopher M. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press.

Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "Siamese" time delay neural network. *International Journal Pattern Recognition and Artificial Intelligence*, 7(4):669–688.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine learning (ICML-05)*, pages 89–96.

Ben Carterette, Paul N. Bennett, David Maxwell Chickering, and Susan Dumais. 2008. Here or there: Preference judgments for relevance. In *Proceedings of the 30th European Conference on Information Retrieval (ECIR 2008)*.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of CVPR-2005*, pages 539–546.

Corinna Cortes and Mehryar Mohri. 2004. AUC optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems (NIPS 2003)*.

Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz Kandola. 2002. On kernel-target alignment. In *Advances in Neural Information Processing Systems 14*, pages 367–373. MIT Press.

Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Ofer Dekel, Christopher D. Manning, and Yoram Singer. 2004. Log-linear models for label ranking. In *Advances in Neural Information Processing Systems (NIPS 2003)*.

Pedro Domingos. 1999. MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of KDD-1999*, pages 155–164.

Joshua Goodman and Vitor R. Carvalho. 2005. Implicit queries for email. In *Proceedings of the 2nd conference on Email and Anti-Spam (CEAS-2005)*.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 2000. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, pages 115–132.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th World Wide Web Conference*.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proc. of COLING-ACL 98*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Pres.

Donald Metzler, Susan Dumais, and Christopher Meek. 2007. Similarity measures for short segments of text. In *Proceedings of the 29th European Conference on Information Retrieval (ECIR 2007)*.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of AAAI-2006*.

Katharina Morik, Peter Brockhausen, and Thorsten Joachims. 1999. Combining statistical learning with a knowledge-based approach – a case study in intensive care monitoring. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-1999)*, pages 268–277.

Jorge Nocedal and Stephen Wright. 2006. *Numerical Optimization*. Springer, 2nd edition.

Mehran Sahami and Timothy D. Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th World Wide Web Conference*.

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. 2008. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceedings of KDD-2008*, pages 614–622.

Wen-tau Yih and Christopher Meek. 2007. Improving similarity measures for short segments of text. In *Proceedings of AAAI-2007*, pages 1489–1494.

Wen-tau Yih, Joshua Goodman, and Vitor Carvalho. 2006. Finding advertising keywords on web pages. In *Proceedings of the 15th World Wide Web Conference*.

A Relational Model of Semantic Similarity between Words using Automatically Extracted Lexical Pattern Clusters from the Web

Danushka Bollegala * danushka@mi.ci.i.u-tokyo.ac.jp
Yutaka Matsuo matsuo@biz-model.t.u-tokyo.ac.jp
Mitsuru Ishizuka ishizuka@i.u-tokyo.ac.jp
The University of Tokyo
7-3-1, Hongo, Tokyo, 113-8656, Japan

Abstract

Semantic similarity is a central concept that extends across numerous fields such as artificial intelligence, natural language processing, cognitive science and psychology. Accurate measurement of semantic similarity between words is essential for various tasks such as, document clustering, information retrieval, and synonym extraction. We propose a novel model of semantic similarity using the semantic relations that exist among words. Given two words, first, we represent the semantic relations that hold between those words using automatically extracted lexical pattern clusters. Next, the semantic similarity between the two words is computed using a Mahalanobis distance measure. We compare the proposed similarity measure against previously proposed semantic similarity measures on Miller-Charles benchmark dataset and WordSimilarity-353 collection. The proposed method outperforms all existing web-based semantic similarity measures, achieving a Pearson correlation coefficient of 0.867 on the Millet-Charles dataset.

1 Introduction

Similarity is a fundamental concept in theories of knowledge and behavior. Psychological experiments have shown that similarity acts as an organizing principle by which individuals classify objects, and make generalizations (Goldstone, 1994). For example, a biologist would classify a newly found animal specimen based upon the properties that it shares with existing categories of animals. We can then make additional inferences on the new specimen using the properties known for the existing category. As the similarity between two objects X and Y increases, so does the probability of correctly inferring that Y has the property T upon knowing that X has T (Tenenbaum, 1999). Accurate measurement of semantic similarity between lexical units such as words or phrases is important for numerous tasks in natural language processing such as word sense disambiguation (Resnik, 1995), synonym extraction (Lin, 1998a), and automatic thesauri generation (Curran, 2002). In information retrieval, similar or related words are used to expand user queries to improve recall (Sahami and Heilman, 2006).

Semantic similarity is a context dependent and dynamic phenomenon. New words are constantly being created and existing words are assigned with new senses on the Web. To decide whether two words are semantically similar, it is important to know the semantic relations that hold between the words. For example, the words *horse* and *cow* can be considered semantically similar because both horses and cows are useful animals in agriculture. Similarly, a *horse* and a *car* can be considered semantically similar because cars, and historically horses, are used for transportation. Semantic relations such as ***X* and *Y* are used in agriculture**, or ***X* and *Y* are used for transportation**, exist between two words *X* and *Y* in these examples. We use bold-italics, ***X***, to denote the slot of a word *X* in a lexical pattern.

We propose a *relational model* to compute the semantic similarity between two words. First, using snippets retrieved from a web search engine, we present an automatic lexical pattern extraction algorithm to represent the semantic relations that exist between two words. For example, given two words *ostrich* and *bird*, we extract ***X* is a *Y***, ***X* is a large *Y***, and ***X* is a flightless *Y*** from the Web. Using a set of semantically related words as training data, we evaluate the confidence of a lexical

*Research Fellow of the Japan Society for the Promotion of Science (JSPS)

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 803–812,
Singapore, 6-7 August 2009.

pattern as an indicator of semantic similarity. For example, the pattern ***X** is a **Y*** is a better indicator of semantic similarity between *X* and *Y* than the pattern ***X** and **Y***. Consequently, we would like to emphasize the former pattern by assigning it a higher confidence score. It is noteworthy that all lexical patterns are not independent – multiple lexical patterns can express the same semantic relation. For example, the pattern ***X** is a large **Y*** subsumes the more general pattern ***X** is a **Y*** and they both indicate a hypernymic relationship between *X* and *Y*. By clustering the semantically related patterns into groups, we can both overcome the data sparseness problem, and reduce the number of parameters during training. To identify semantically related patterns, we use a sequential pattern clustering algorithm that is based on the distributional hypothesis (Harris, 1954). We represent two words by a feature vector defined over the clusters of patterns. Finally, the semantic similarity is computed as the Mahalanobis distance between points corresponding to the feature vectors. By using Mahalanobis distance instead of Euclidean distance, we can account for the inter-dependence between semantic relations.

2 Related Work

Geometric models, such as multi-dimensional scaling has been used in psychological experiments analyzing the properties of similarity (Krumhansl, 1978). These models represent objects as points in some coordinate space such that the observed dissimilarities between objects correspond to the metric distances between the respective points. Geometric models assume that objects can be adequately represented as points in some coordinate space and that dissimilarity behaves like a metric distance function satisfying minimality, symmetry, and triangle inequality assumptions. However, both dimensional and metric assumptions are open to question.

Tversky (1977) proposed the *contrast model* of similarity to overcome the problems in geometric models. The contrast model relies on featural representation of objects, and it is used to compute the similarity between the representations of two objects. Similarity is defined as an increasing function of common features (i.e. features in common to the two objects), and as a decreasing function of distinctive features (i.e. features that apply to one object but not the other). The attributes of objects are primal to contrast model and it does not explicitly incorporate the relations between objects when measuring similarity.

Hahn et al. (2003) define similarity between two representations as the complexity required to transform one representation into the other. Their model of similarity is based on the *Representational Distortion* theory, which aims to provide a theoretical framework of similarity judgments. Their experiments using pattern sequences and geometric shapes show an inverse correlation between the number of transformations required to convert one pattern (or shape) to another, and the perceived similarity ratings by human subjects. How to represent an object, which transformations are allowed on a representation, and how to measure the complexity of a transformation, are all important decisions in the transformational model of similarity. Although distance measures such as edit distance have been used to find approximate matches in a dictionary, it is not obvious how to compute semantic similarity between words using representational distortion theory.

Given a taxonomy of concepts, a straightforward method to calculate similarity between two words (or concepts) is to find the length of the shortest path connecting the two words in the taxonomy (Rada et al., 1989). If a word is polysemous (i.e. has more than one sense) then multiple paths might exist between the two words. In such cases, only the shortest path between any two senses of the words is considered for calculating similarity. A problem that is frequently acknowledged with this approach is that it relies on the notion that all links in the taxonomy represent a uniform distance. As a solution to this problem, Schickel-Zuber and Faltings (2007) propose ontology structure based similarity (OSS) between two concepts in an ontology, which is an asymmetric distance function.

Resnik (1995) proposed a similarity measure using information content. He defined the similarity between two concepts C_1 and C_2 in the taxonomy as the maximum of the information content of all concepts C that subsume both C_1 and C_2. Then the similarity between two words is defined as the maximum of the similarity between any concepts that the words belong to. He used WordNet as the taxonomy; information content is calculated using the Brown corpus.

Li et al., (2003) combined structural seman-

tic information from a lexical taxonomy, and information content from a corpus, in a nonlinear model. They proposed a similarity measure that uses shortest path length, depth and local density in a taxonomy. Their experiments reported a Pearson correlation coefficient of 0.8914 on the Miller-Charles benchmark dataset (Miller and Charles, 1998). Lin (1998b) defined the similarity between two concepts as the information that is in common to both concepts and the information contained in each individual concept.

Cilibrasi and Vitanyi (2007) proposed a distance metric between words using page-counts retrieved from a web search engine. The proposed metric is named *Normalized Google Distance* (NGD) and is defined as the normalized information distance (Li et al., 2004) between two strings. They evaluate NGD in a word classification task. Unfortunately NGD only uses page-counts of words and ignores the context in which the words appear. Therefore, it produces inaccurate similarity scores when one or both words between which similarity is computed are polysemous.

Sahami and Heilman (2006) measured semantic similarity between two queries using snippets returned for those queries by a search engine. For each query, they collect snippets from a search engine and represent each snippet as a TF-IDF-weighted term vector. Each vector is L_2 normalized and the centroid of the set of vectors is computed. Semantic similarity between two queries is then defined as the inner product between the corresponding centroid vectors. They did not compare their similarity measure with taxonomy-based similarity measures.

Chen et al., (2006) propose a web-based double-checking model to compute the semantic similarity between words. For two words X and Y, they collect snippets for each word from a web search engine. Then they count the number of occurrences of X in the snippets for Y, and Y in the snippets for X. The two values are combined nonlinearly to compute the similarity between X and Y. This method heavily depends on the search engine's ranking algorithm. Although two words X and Y may be very similar, there is no reason to believe that one can find Y in the snippets for X, or vice versa. This observation is confirmed by the experimental results in their paper which reports 0 similarity scores for many pairs of words in the Miller-Charles dataset.

In our previous work (Bollegala et al., 2007), we proposed a semantic similarity measure using page counts and snippets retrieved from a Web search engine. To compute the similarity between two words X and Y, we queried a web search engine using the query *X AND Y* and extract lexical patterns that combine X and Y from snippets. A feature vector is formed using frequencies of 200 lexical patterns in snippets and four co-occurrence measures: Dice coefficient, overlap coefficient, Jaccard coefficient and pointwise mutual information. We trained a two-class support vector machine using automatically selected synonymous and non-synonymous word pairs from WordNet. This method reports a Pearson correlation coefficient of 0.837 with Miller-Charles ratings. However, it does not consider the relatedness between patterns.

Gabrilovich and Markovitch (2007) represent words using weighted vectors of Wikipedia-based concepts, and define the similarity between words as the cosine of the angle between the corresponding vectors. Their method can be used to compute similarity between words as well as between texts. Although Wikipedia is growing in popularity, not all concepts found on the Web have articles in Wikipedia. Specially, novel or not very popular concepts are not adequately covered by Wikipedia. Moreover, their method requires the concepts to be independent. For non-independent, hierarchical taxonomies such as open directory project (ODP)[1], their method produces suboptimal results.

3 Relational Model of Similarity

We propose a model to compute the semantic similarity between two words a and b using the set of semantic relations $R(a, b)$ that hold between a and b. We call the proposed model the *relational model* of semantic similarity and it is defined by the following equation,

$$\mathrm{sim}(a, b) = \Xi(R(a, b)). \quad (1)$$

Here, $\mathrm{sim}(a, b)$ is the semantic similarity between the two words a and b, and Ξ is a weighting function defined over the set of semantic relations $R(a, b)$. Given that a particular set of semantic relations are known to hold between two words, the function Ξ expresses our confidence on those words being semantically similar.

[1] http://www.dmoz.org

A semantic relation can be expressed in a number of ways. For example, given a taxonomy of words such as the WordNet, semantic relations (i.e. hypernymy, meronymy, synonymy etc.) between words can be directly looked up in the taxonomy. Alternatively, the labels of the edges in the path connecting two words can be used as semantic relations. However, in this paper we do not assume the availability of manually created resources such as dictionaries or taxonomies. We represent semantic relations using automatically extracted lexical patterns. Lexical patterns have been successfully used to represent various semantic relations between words such as hypernymy (Hearst, 1992), and meronymy (Berland and Charniak, 1999). Following these previous approaches, we represent $R(a, b)$ as a set of lexical patterns. Moreover, we denote the frequency of a lexical pattern r for a word pair (a, b) by $f(r, a, b)$.

So far we have not defined the functional form of Ξ. A straightforward approach is to use a linearly weighted combination of relations as shown below,

$$\Xi(R(a,b)) = \sum_{r_i \in R(a,b)} w_i \times f(r_i, a, b). \quad (2)$$

Here, w_i is the weight associated with the lexical pattern r_i and can be determined using training data. However, this formulation has two fundamental drawbacks. First, the number of weight parameters w_i is equal to the number of lexical patterns. Typically two words can co-occur in numerous patterns. Consequently, we end up with a large number of parameters in the model. Complex models with a large number of parameters are difficult to train because they tend to overfit to the training data. Second, the linear combination given in Equation 2 assumes the lexical patterns to be mutually independent. However, in practice this is not true. For example, both patterns *X is a Y* and *Y such as X* indicate a hypernymic relation between *X* and *Y*.

To overcome the above mentioned limitations, we first cluster the lexical patterns to identify the semantically related patterns. Our clustering algorithm is detailed in section 3.2. Next, we define Ξ using the formed clusters as follows,

$$\Xi(R(a,b)) = \mathbf{x}_{ab}^T \Lambda \sigma. \quad (3)$$

Here, $\mathbf{x}_{ab}$ is a feature vector representing the words a and b. Each formed cluster contributes a feature in vector $\mathbf{x}_{ab}$ as described later in Section 5. The vector σ is a prototypical vector representing synonymous word pairs. We compute σ as the centroid of feature vectors representing synonymous word pairs. Λ is the inter-cluster correlation matrix. The (i, j)-th element of matrix Λ denotes the correlation between the two clusters c_i and c_j. Matrix Λ is expected to capture the dependence between semantic relations. Intuitively, if two clusters i and j are highly correlated, then the (i, j)-th element of Λ will be closer to 1. Equation 3 computes the similarity between a word pair (a, b) and a set of synonymous word pairs. Intuitively, if the relations that exist between a and b are typical relations that hold between synonymous word pairs, then Equation 3 returns a high similarity score for a and b.

The proposed relational model of semantic similarity differs from feature models of similarity, such as the contrast model (Tversky, 1977), in that it is defined over the set of semantic relations that exist between two words instead of the set of features for each word. Specifically, in contrast model, the similarity $S(a, b)$ between two objects a and b is defined in terms of the features common to a and b, $A \cap B$, the features that are distinctive to a, $A - B$, and the features that are distinctive to b, $B - A$. The contrast model is formalized in the following equation,

$$S(a,b) = \theta f(A \cap B) - \alpha f(A - B) - \beta f(B - A). \quad (4)$$

Here, the function f measures the salience of a particular set of features, and non-negative parameters α, β, and θ determine the relative weights assigned to the different components. However, in the relational model of similarity we do not focus on features of individual words but on relations between two words.

Modeling similarity as a phenomenon of relations between objects rather than features of individual objects is central to computational models of analogy-making such as the structure mapping theory (SMT) (Falkenhainer et al., 1989). SMT claims that an analogy is a mapping of knowledge from one domain (base) into another (target) which conveys that a system of relations known to hold in the base also holds in the target. The target objects do not have to resemble their corresponding base objects. During the mapping process, features of individual objects are dropped and only relations are mapped. The proposed relational model of similarity uses this relational view

Ostrich, a large, flightless bird that lives in the dry grasslands of Africa.

Figure 1: A snippet returned for the query *"ostrich * * * * * bird"*.

of similarity to compute semantic similarity between words.

3.1 Extracting Lexical Patterns

To compute semantic similarity between two words using the relational model (Equation 3), we must first extract the numerous lexical patterns from contexts in which those two words appear. For this purpose, we propose a pattern extraction algorithm using snippets retrieved from a web search engine. The proposed method requires no language-dependent preprocessing such as part-of-speech tagging or dependency parsing, which can be both time consuming at Web scale, and likely to produce incorrect results because of the fragmented and ill-formed snippets.

Given two words a and b, we query a web search engine using the wildcard query *"a * * * * * b"* and download snippets. The "*" operator matches one word or none in a web page. Therefore, our wildcard query retrieves snippets in which a and b appear within a window of seven words. We attempt to approximate the local context of two words using wildcard queries. For example, Figure 1 shows a snippet retrieved for the query *"ostrich * * * * * bird"*.

For a snippet S, retrieved for a word pair (a, b), first, we replace the two words a and b, respectively, with two variables *X* and *Y*. We replace all numeric values by D, a marker for digits. Next, we generate all subsequences of words from S that satisfy all of the following conditions.

(i). A subsequence must contain exactly one occurrence of each *X* and *Y*

(ii). The maximum length of a subsequence is L words.

(iii). A subsequence is allowed to have gaps. However, we do not allow gaps of more than g number of words. Moreover, the total length of all gaps in a subsequence should not exceed G words.

(iv). We expand all negation contractions in a context. For example, *didn't* is expanded to *did not*. We do not skip the word *not* when generating subsequences. For example, this condition ensures that from the snippet *X is not a Y*, we do not produce the subsequence *X is a Y*.

Finally, we count the frequency of all generated subsequences and only use subsequences that occur more than N times as lexical patterns.

The parameters *L, g, G* and *N* are set experimentally, as explained later in Section 6. It is noteworthy that the proposed pattern extraction algorithm considers all the words in a snippet, and is *not* limited to extracting patterns only from the mid-fix (i.e., the portion of text in a snippet that appears between the queried words). Moreover, the consideration of gaps enables us to capture relations between distant words in a snippet. We use a modified version of the *prefixspan* algorithm (Pei et al., 2004) to generate subsequences from a text snippet. Specifically, we use the constraints (ii)-(iv) to prune the search space of candidate subsequences. For example, if a subsequence has reached the maximum length L, or contains the maximum number of gaps G, then we will not extend it further. By pruning the search space, we can speed up the pattern generation process. However, none of these modifications affect the accuracy of the proposed semantic similarity measure because the modified version of the prefixspan algorithm still generates the exact set of patterns that we would obtain if we used the original prefixspan algorithm (i.e. without pruning) and subsequently remove patterns that violate the above mentioned constraints. For example, some patterns extracted form the snippet shown in Figure 1 are: ***X**, a large **Y***, ***X** a flightless **Y***, and ***X**, large **Y** lives*.

3.2 Clustering Lexical Patterns

A semantic relation can be expressed using more than one pattern. By grouping the semantically related patterns, we can both reduce the model complexity in Equation 2, and consider the dependence among semantic relations in Equation 3. We use the distributional hypothesis (Harris, 1954) to find semantically related lexical patterns. The distributional hypothesis states that words that occur in the same context have similar meanings. If two lexical patterns are similarly distributed over a set of word pairs, then from the distributional hypothesis it follows that the two patterns must be similar.

We represent a pattern p by a vector $\mathbf{p}$ in which

the i-th element is the frequency $f(a_i, b_i, p)$ of p in a word pair (a_i, b_i). Given a set P of patterns and a similarity threshold θ, Algorithm 1 returns clusters of similar patterns. First, the function $SORT$ sorts the patterns in the descending order of their total occurrences in all word pairs. The total occurrences of a pattern p is defined as $\mu(p)$, and is given by,

$$\mu(p) = \sum_{(a,b) \in W} f(a, b, p). \quad (5)$$

Here, W is the set of word pairs. Then the outer for-loop (starting at line 3), repeatedly takes a pattern $\mathbf{p_i}$ from the ordered set P, and in the inner for-loop (starting at line 6), finds the cluster, c^* ($\in C$) that is most similar to $\mathbf{p_i}$. Similarity between $\mathbf{p_i}$ and the cluster centroid $\mathbf{c_j}$ is computed using cosine similarity. The centroid vector $\mathbf{c}_j$ of cluster c_j is defined as the vector sum of all pattern vectors for patterns in that cluster (i.e. $\mathbf{c}_j = \sum_{p \in c_j} \mathbf{p}$). If the maximum similarity exceeds the threshold θ, we append $\mathbf{p_i}$ to $\mathbf{c}^*$ (line 14). Here, the operator $\oplus$ denotes vector addition. Otherwise, we form a new cluster $\{\mathbf{p_i}\}$ and append it to C, the set of clusters. After all patterns are clustered, we compute the (i, j) element of the inter-cluster correlation matrix Λ (Equation 3) as the inner-product between the centroid vectors $\mathbf{c_i}$ and $\mathbf{c_j}$ of the corresponding clusters i and j. The parameter θ ($\in [0, 1]$) determines the *purity* of the formed clusters and is set experimentally in Section 5. Algorithm 1 scales linearly with the number of patterns. Moreover, sorting the patterns by their total word pair frequency prior to clustering ensures that the final set of clusters contains the most common relations in the dataset.

Algorithm 1 Sequential pattern clustering algorithm.

Input: patterns $P = \{\mathbf{p_1}, \ldots, \mathbf{p_n}\}$, threshold θ
Output: clusters C

```
1:  SORT(P)
2:  C ← {}
3:  for pattern p_i ∈ P do
4:      max ← −∞
5:      c* ← null
6:      for cluster c_j ∈ C do
7:          sim ← cosine(p_i, c_j)
8:          if sim > max then
9:              max ← sim
10:             c* ← c_j
11:         end if
12:     end for
13:     if max ≥ θ then
14:         c* ← c* ⊕ p_i
15:     else
16:         C ← C ∪ {p_i}
17:     end if
18: end for
19: return C
```

4 Evaluation Procedure

Evaluating a semantic similarity measure is difficult because the notion of semantic similarity is subjective. Miller-Charles (1998) dataset has been frequently used to benchmark semantic similarity measures. Miller-Charles dataset contains 30 word pairs rated by a group of 38 human subjects. The word pairs are rated on a scale from 0 (no similarity) to 4 (perfect synonymy). Because of the omission of two word pairs in earlier versions of WordNet, most researchers had used only 28 pairs for evaluations. The degree of correlation between the human ratings in the benchmark dataset and the similarity scores produced by an automatic semantic similarity measure, can be considered as a measurement of how well the semantic similarity measure captures the notion of semantic similarity held by humans. In addition to Miller-Charles dataset we also evaluate on the WordSimilarity-353 (Finkelstein et al., 2002) dataset. In contrast to Miller-Charles dataset which has only 30 word pairs, WordSimilarity-353 dataset contains 353 word pairs. Each pair has 13-16 human judgments, which were averaged for each pair to produce a single relatedness score. Following the previous work, we use both Miller-Charles dataset and WordSimilarity-353 dataset to evaluate the proposed semantic similarity measure.

5 Computing Semantic Similarity

To extract lexical patterns that express numerous semantic relations, we first select synonymous words from WordNet synsets. A synset is a set of synonymous words assigned for a particular sense of a word in WordNet. We randomly select 2000 synsets of nouns from WordNet. From each synset, a pair of synonymous words is selected. For polysemous nouns, we selected synonyms from the dominant sense. To perform a fair evaluation, we do not select any words that appear in the Miller-Charles dataset or the WordSimilarity-353

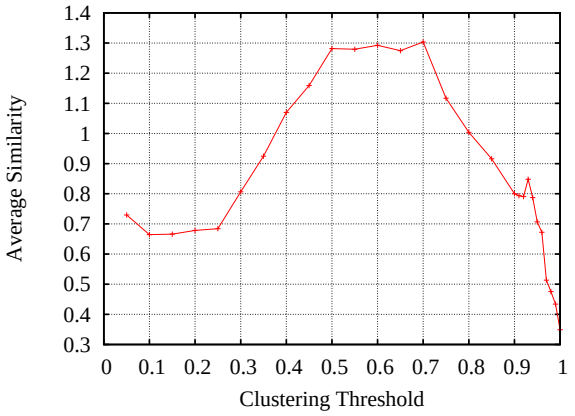


Figure 2: Average similarity vs. clustering threshold θ

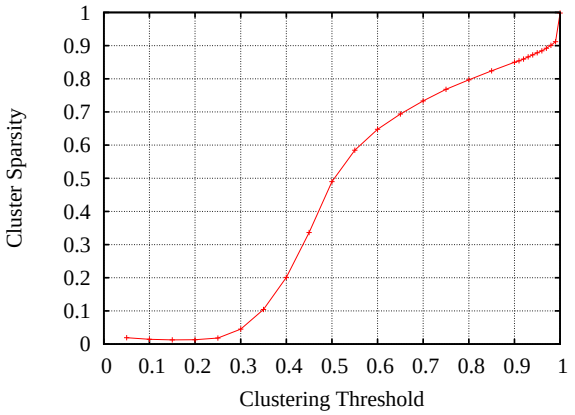


Figure 3: Sparsity vs. clustering threshold θ

dataset, which are used later for evaluation purposes. As we describe later, the clustering threshold θ is tuned using this set of 2000 word pairs selected from the WordNet.

We use the YahooBOSS API[2] and download 1000 snippets for each of those word pairs. Experimentally, we set the values for the parameters in the pattern extraction algorithm (Section 3.1): $L = 5$, $g = 2$, $G = 4$, and extract $5,238,637$ unique patterns. However, only $1,680,914$ of those patterns occur more than twice. Low frequency patterns often contain misspellings and are not suitable for training. Therefore, we selected patterns that occur at least 10 times in the snippet collection. Moreover, we remove very long patterns (ca. over 20 characters). The final set contains $140,691$ unique lexical patterns. The remainder of the experiments described in the paper use those patterns.

[2]http://developer.yahoo.com/search/boss/

We use the clustering Algorithm 1 to cluster the extracted patterns. The only parameter in Algorithm 1, the clustering threshold θ, is set as follows. We vary the value of theta θ from 0 to 1, and use Algorithm 1 to cluster the extracted set of patterns. We use the resultant set of clusters to represent a word pair by a feature vector. We compute a feature from each cluster as follows. First, we assign a weight w_{ij} to a pattern p_i that is in a cluster c_j as follows,

$$w_{ij} = \frac{\mu(p_i)}{\sum_{q \in c_j} \mu(q)}. \quad (6)$$

Here, $\mu(q)$ is the total frequency of a pattern, and it is given by Equation 5. Because we perform a hard clustering on patterns, a pattern can belong to only one cluster (i.e. $w_{ij} = 0$ for $p_i \notin c_j$). Finally, we compute the value of the j-th feature in the feature vector for word pair (a, b) as follows,

$$\sum_{p_i \in c_j} w_{ij} f(a, b, p_i). \quad (7)$$

For each set of clusters, we compute the element Λ_{ij} of the corresponding inter-cluster correlation matrix Λ by the cosine similarity between the centroid vectors for clusters c_i and c_j. The prototype vector σ in Equation 3 is computed as the vector sum of individual feature vectors for the synonymous word pairs selected from the WordNet as described above. We then use Equation 3 to compute the average of similarity scores for synonymous word pairs we selected from WordNet.

We select the θ that maximizes the average similarity score between those synonymous word pairs. Formally, the optimal value of θ, $\hat{\theta}$ is given by the following Equation,

$$\hat{\theta} = \text{argmax}_{\theta \in [0,1]} \left(\frac{1}{|W|} \sum_{(a,b) \in W} \text{sim}(a, b) \right). \quad (8)$$

Here, W is the set of synonymous word pairs (a, b), $|W|$ is the total number of synonymous word pairs (i.e. 2000 in our experiments), and $\text{sim}(a, b)$ is given by Equation 3. Because the averages are taken over 2000 word pairs this procedure gives a reliable estimate for θ. Moreover, this method does not require negative training instances such as, non-synonymous word pairs, which are difficult to create manually. Average similarity scores for various θ values are shown in Figure 2. From Figure 2, we see that initially average similarity increases when θ is increased.

This is because clustering of semantically related patterns reduces the sparseness in feature vectors. Average similarity is stable within a range of θ values between 0.5 and 0.7. However, increasing θ beyond 0.7 results in a rapid drop of average similarity. To explain this behavior consider Figure 3 where we plot the sparsity of the set of clusters (i.e. the ratio between singletons to total clusters) against threshold θ. As seen from Figure 3, high θ values result in a high percentage of singletons because only highly similar patterns will form clusters. Consequently, feature vectors for different word pairs do not have many features in common. The maximum average similarity score of 1.303 is obtained with $\theta = 0.7$, corresponding to $17,015$ total clusters out of which $12,476$ are singletons with exactly one pattern (sparsity = 0.733). For the remainder of the experiments in this paper we set θ to this optimal value and use the corresponding set of clusters to compute semantic similarity by Equation 3. Similarity scores computed using Equation 3 can be greater than 1 (see Figure 2) because of the terms corresponding to the non-diagonal elements in Λ. We do not normalize the similarity scores to $[0, 1]$ range in our experiments because the evaluation metrics we use are insensitive to linear transformations of similarity scores.

6 Experiments

Table 1 compares the proposed method against Miller-Charles ratings (MC), and previously proposed web-based semantic similarity measures: Jaccard, Dice, Overlap, PMI (Bollegala et al., 2007), Normalized Google Distance (NGD) (Cilibrasi and Vitanyi, 2007), Sahami and Heilman (SH) (2006), co-occurrence double checking model (CODC) (Chen et al., 2006), and support vector machine-based (SVM) approach (Bollegala et al., 2007). The bottom row of Table 1 shows the Pearson correlation coefficient of similarity scores produced by each algorithm with MC. All similarity scores, except for the human-ratings in Miller-Charles dataset, are normalized to $[0, 1]$ range for the ease of comparison. It is noteworthy that the Pearson correlation coefficient is invariant under a linear transformation. All similarity scores shown in Table 1 except for the proposed method are taken from the original published papers.

The highest correlation is reported by the proposed semantic similarity measure. The improvement of the proposed method is statistically significant (confidence interval $[0.73, 0.93]$) against all the similarity measures compared in Table 1 except against the SVM approach. From Table 1 we see that measures that use contextual information from snippets (e.g. SH, CODC, SVM, and proposed) outperform the ones that use only co-occurrence statistics (e.g. Jaccard, overlap, Dice, PMI, and NGD) such as page-counts. This is because similarity measures that use contextual information are better equipped to compute the similarity between polysemous words. Although both SVM and proposed methods use lexical patterns, unlike the proposed method, the SVM method does not consider the relatedness between patterns. The superior performance of the proposed method is attributable to its consideration of relatedness of patterns.

Table 2 summarizes the previously proposed WordNet-based semantic similarity measures. Despite the fact that the proposed method does not use manually compiled resources such as WordNet for computing similarity, its performance is comparable to similarity measures that use WordNet. We believe that the proposed method will be useful to compute the semantic similarity between named-entities for which manually created resources are either incomplete or do not exist.

We evaluate the proposed method using the WordSimilarity-353 dataset. Experimental results are presented in Table 3. Following previous work, we use Spearman rank correlation coefficient, which does not require ratings to be linearly dependent, for the evaluations on this dataset. Likewise with the Miller-Charles ratings, we measure the correlation between the similarity scores produced by the proposed method for word pairs in the WordSimilarity-353 dataset and the human ratings. A higher Spearman correlation coefficient (value=0.504, confidence interval $[0.422, 0.578]$) indicates a better agreement with the human notion of semantic similarity. From Table 3 we can see that the proposed method outperforms a wide variety of semantic similarity measures developed using numerous resources including lexical resources such as WordNet and knowledge sources such as Wikipedia (i.e. WikiRelate!). In contrast to the Miller-Charles dataset which only contains common English words selected from the WordNet, the WordSimilarity-353 dataset contains word pairs where one or both words are named entities (e.g. (*Maradona*, *foot-*

Table 1: Semantic similarity scores on Miller-Charles dataset

Word Pair	MC	Jaccrad	Dice	Overlap	PMI	NGD	SH	CODC	SVM	Proposed
automobile-car	3.920	0.650	0.664	0.831	0.427	0.466	0.225	0.008	0.980	0.918
journey-voyage	3.840	0.408	0.424	0.164	0.468	0.556	0.121	0.005	0.996	1.000
gem-jewel	3.840	0.287	0.300	0.075	0.688	0.566	0.052	0.012	0.686	0.817
boy-lad	3.760	0.177	0.186	0.593	0.632	0.456	0.109	0.000	0.974	0.958
coast-shore	3.700	0.783	0.794	0.510	0.561	0.603	0.089	0.006	0.945	0.975
asylum-madhouse	3.610	0.013	0.014	0.082	0.813	0.782	0.052	0.000	0.773	0.794
magician-wizard	3.500	0.287	0.301	0.370	0.863	0.572	0.057	0.008	1.000	0.997
midday-noon	3.420	0.096	0.101	0.116	0.586	0.687	0.069	0.010	0.819	0.987
furnace-stove	3.110	0.395	0.410	0.099	1.000	0.638	0.074	0.011	0.889	0.878
food-fruit	3.080	0.751	0.763	1.000	0.449	0.616	0.045	0.004	0.998	0.940
bird-cock	3.050	0.143	0.151	0.144	0.428	0.562	0.018	0.006	0.593	0.867
bird-crane	2.970	0.227	0.238	0.209	0.516	0.563	0.055	0.000	0.879	0.846
implement-tool	2.950	1.000	1.000	0.507	0.297	0.750	0.098	0.005	0.684	0.496
brother-monk	2.820	0.253	0.265	0.326	0.623	0.495	0.064	0.007	0.377	0.265
crane-implement	1.680	0.061	0.065	0.100	0.194	0.559	0.039	0.000	0.133	0.056
brother-lad	1.660	0.179	0.189	0.356	0.645	0.505	0.058	0.005	0.344	0.132
car-journey	1.160	0.438	0.454	0.365	0.205	0.410	0.047	0.004	0.286	0.165
monk-oracle	1.100	0.004	0.005	0.002	0.000	0.579	0.015	0.000	0.328	0.798
food-rooster	0.890	0.001	0.001	0.412	0.207	0.568	0.022	0.000	0.060	0.018
coast-hill	0.870	0.963	0.965	0.263	0.350	0.669	0.070	0.000	0.874	0.356
forest-graveyard	0.840	0.057	0.061	0.230	0.495	0.612	0.006	0.000	0.547	0.442
monk-slave	0.550	0.172	0.181	0.047	0.611	0.698	0.026	0.000	0.375	0.243
coast-forest	0.420	0.861	0.869	0.295	0.417	0.545	0.060	0.000	0.405	0.150
lad-wizard	0.420	0.062	0.065	0.050	0.426	0.657	0.038	0.000	0.220	0.231
cord-smile	0.130	0.092	0.097	0.015	0.208	0.460	0.025	0.000	0	0.006
glass-magician	0.110	0.107	0.113	0.396	0.598	0.488	0.037	0.000	0.180	0.050
rooster-voyage	0.080	0.000	0.000	0.000	0.228	0.487	0.049	0.000	0.017	0.052
noon-string	0.080	0.116	0.123	0.040	0.102	0.488	0.024	0.000	0.018	0.000
Correlation	**-**	**0.260**	**0.267**	**0.382**	**0.549**	**0.205**	**0.580**	**0.694**	**0.834**	**0.867**

Table 2: Comparison with WordNet-based similarity measures.

Method	Correlation
Edge-counting	0.664
Jiang & Conrath (1998)	0.848
Lin (1998a)	0.822
Resnik (1995)	0.745
Li et al. (2003)	0.891

Table 3: Results on WordSimilarity-353 dataset.

Method	Correlation
WordNet Edges (Jarmasz, 1993)	0.27
Hirst & St-Onge (1997)	0.34
Jiang & Conrath (1998)	0.34
WikiRelate! (Strube and Ponzetto, 2006)	0.19-0.48
Leacock & Chodrow (1998)	0.36
Lin (1998b)	0.36
Resnik (1995)	0.37
Proposed	0.504

ball) and (*Jerusalem*, *Israel*)). Because the proposed method use snippets retrieved from a web search engine, it is capable of extracting expressive lexical patterns that can explicitly state the relationship between two entities.

If we must compare n objects using a feature model of similarity, then we only need to define features for each of those n objects. However, in the proposed relational model we must define relations between all pairs of objects. In the case where all n objects are different, this requires us to define relations for $n(n-1)/2$ object pairs. Defining relations for all pairs can be computationally costly for large n values. Efficiently comparing n objects using a relational model is an interesting future research direction of the current work.

7 Conclusion

We proposed a relational model to measure the semantic similarity between two words. First, to represent the numerous semantic relations that exist between two words, we extract lexical patterns from snippets retrieved from a web search engine. Second, we cluster the extracted patterns to identify the semantically related patterns. Third, using the pattern clusters we define a feature vector to represent two words and compute the semantic similarity by taking into account the inter-cluster correlation. The proposed method outperformed all existing web-based semantic similarity measures on two benchmark datasets.

References

M. Berland and E. Charniak. 1999. Finding parts in very large corpora. In *Proc. of ACL'99*, pages 57–64.

D. Bollegala, Y. Matsuo, and M. Ishizuka. 2007. Measuring semantic similarity between words using web search engines. In *Proc. of WWW'07*, pages 757–766.

H. Chen, M. Lin, and Y. Wei. 2006. Novel association measures using web search with double checking. In *Proc. of the COLING/ACL '06*, pages 1009–1016.

R.L. Cilibrasi and P.M.B. Vitanyi. 2007. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383.

J. Curran. 2002. Ensemble menthods for automatic thesaurus extraction. In *Proc. of EMNLP*.

B. Falkenhainer, K.D. Forbus, and D. Gentner. 1989. Structure mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. 2002. Placing search in context: The concept revisited. *ACM TOIS*, 20:116–131.

E. Gabrilovich and S. Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proc. of IJCAI'07*, pages 1606–1611.

R. L. Goldstone. 1994. The role of similarity in categorization: providing a groundwork. *Cognition*, 52:125–157.

U. Hahn, N. Chater, and L. B. Richardson. 2003. Similarity as transformation. *Cognition*, 87:1–32.

Z. Harris. 1954. Distributional structure. *Word*, 10:146–162.

M.A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of 14th COLING*, pages 539–545.

G. Hirst and D. St-Onge. 1997. Lexical chains as representations of context for the detection and correction of malapropisms.

M. Jarmasz. 1993. Roget's thesaurus as a lexical resource for natural language processing. Master's thesis, University of Ottawa.

J.J. Jiang and D.W. Conrath. 1998. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of ROCLING'98*.

C. L. Krumhansl. 1978. Concerning the applicability of geometric models to similarity data: The interrelationship between similarity and spatial density. *Psychological Review*, 85:445–463.

C. Leacock and M. Chodorow. 1998. *Combining Local Context and WordNet Similarity for Word Sense Identification*. MIT.

M. Li, X. Chen, X. Li, B. Ma, and P.M.B. Vitanyi. 2004. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264.

D. Lin. 1998a. Automatic retreival and clustering of similar words. In *Proc. of the 17th COLING*, pages 768–774.

D. Lin. 1998b. An information-theoretic definition of similarity. In *Proc. of the 15th ICML*, pages 296–304.

G. Miller and W. Charles. 1998. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

J. Pei, J. Han, B. Mortazavi-Asi, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. 2004. Mining sequential patterns by pattern-growth: the prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440.

R. Rada, H. Mili, E. Bichnell, and M. Blettner. 1989. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):17–30.

P. Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proc. of IJCAI'95*.

M. Sahami and T. Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proc. of WWW'06*.

V. Schickel-Zuber and B. Faltings. 2007. Oss: A semantic similarity function based on hierarchical ontologies. In *Proc. of IJCAI'07*, pages 551–556.

M. Strube and S. P. Ponzetto. 2006. Wikirelate! computing semantic relatedness using wikipedia. In *Proc. of AAAI' 06*.

J. B. Tenenbaum. 1999. Bayesian modeling of human concept learning. In *NIPS'99*.

A. Tversky. 1977. Features of similarity. *Psychological Review*, 84:327–652.

D. McLean Y. Li, Zuhair A. Bandar. 2003. An approch for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):871–882.

Unbounded Dependency Recovery for Parser Evaluation

Laura Rimell and **Stephen Clark**
University of Cambridge
Computer Laboratory
laura.rimell@cl.cam.ac.uk
stephen.clark@cl.cam.ac.uk

Mark Steedman
University of Edinburgh
School of Informatics
steedman@inf.ed.ac.uk

Abstract

This paper introduces a new parser evaluation corpus containing around 700 sentences annotated with unbounded dependencies, from seven different grammatical constructions. We run a series of off-the-shelf parsers on the corpus to evaluate how well state-of-the-art parsing technology is able to recover such dependencies. The overall results range from 25% accuracy to 59%. These low scores call into question the validity of using Parseval scores as a general measure of parsing capability. We discuss the importance of parsers being able to recover unbounded dependencies, given their relatively low frequency in corpora. We also analyse the various errors made on these constructions by one of the more successful parsers.

1 Introduction

Statistical parsers are now obtaining Parseval scores of over 90% on the WSJ section of the Penn Treebank (Bod, 2003; Petrov and Klein, 2007; Huang, 2008; Carreras et al., 2008). McClosky et al. (2006) report an F-score of 92.1% using self-training applied to the reranker of Charniak and Johnson (2005). Such scores, in isolation, may suggest that statistical parsing is close to becoming a solved problem, and that further incremental improvements will lead to parsers becoming as accurate as POS taggers.

A single score in isolation can be misleading, however, for a number of reasons. First, the single score is an aggregate over a highly skewed distribution of all constituent types; evaluations which look at individual constituent or dependency types show that the accuracies on some, semantically important, constructions, such as coordination and PP-attachment, are much lower (Collins, 1999). Second, it is well known that the accuracy of parsers trained on the Penn Treebank degrades when they are applied to different genres and domains (Gildea, 2001). Finally, some researchers have argued that the Parseval metrics (Black et al., 1991) are too forgiving with respect to certain errors and that an evaluation based on syntactic dependencies, for which scores are typically lower, is a better test of parser performance (Lin, 1995; Carroll et al., 1998).

In this paper we focus on the first issue, that the performance of parsers on some constructions is much lower than the overall score. The constructions that we focus on are various unbounded dependency constructions. These are interesting for parser evaluation for the following reasons: one, they provide a strong test of the parser's knowledge of the grammar of the language, since many instances of unbounded dependencies are difficult to recover using shallow techniques in which the grammar is only superficially represented; and two, recovering these dependencies is necessary to completely represent the underlying predicate-argument structure of a sentence, useful for applications such as Question Answering and Information Extraction.

To give an example of the sorts of constructions we are considering, and the (in)ability of parsers to recover the corresponding unbounded dependencies, none of the parsers that we have tested were able to recover the dependencies shown in bold from the following sentences:

We have also developed techniques for recognizing and locating underground nuclear tests through the **waves** *in the ground which they* **generate**.

By Monday , they hope to have a sheaf of **documents** *both sides can* **trust**.

By means of charts showing wave-travel times and depths in the ocean at various locations , it is possible to estimate the **rate** *of approach and probable* **time** *of arrival at Hawaii* **of** *a tsunami getting under way at any spot in the Pacific .*

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 813–821,
Singapore, 6-7 August 2009.

The contributions of this paper are as follows. First, we present the first set of results for the recovery of a variety of unbounded dependencies, for a range of existing parsers. Second, we describe the creation of a publicly available unbounded dependency test suite, and give statistics summarising properties of these dependencies in naturally occurring text. Third, we demonstrate that performing the evaluation is surprisingly difficult, because of different conventions across the parsers as to how the underlying grammar is represented. Fourth, we show that current parsing technology is very poor at representing some important elements of the argument structure of sentences, and argue for a more focused construction-based parser evaluation as a complement to existing grammatical relation-based evaluations. We also perform an error-analysis for one of the more successful parsers.

There has been some prior work on evaluating parsers on long-range dependencies, but no work we are aware of that has the scope and focus of this paper. Clark et al. (2004) evaluated a CCG parser on a small corpus of object extraction cases. Johnson (2002) began the body of work on inserting traces into the output of Penn Treebank (PTB) parsers, followed by Levy and Manning (2004), among others. This PTB work focused heavily on the representation in the Treebank, evaluating against patterns in the trace annotation. In this paper we have tried to be more "formalism-independent" and construction focused.

2 Unbounded Dependency Corpus

2.1 The constructions

An unbounded dependency construction contains a word or phrase which appears to have been moved, while being interpreted in the position of the resulting "gap". An unlimited number of clause boundaries may intervene between the moved element and the gap (hence "unbounded").

The seven constructions in our corpus were chosen for being relatively frequent in text, compared to other unbounded dependency types, and relatively easy to identify. An example of each construction, along with its associated dependencies, is shown in Table 1. Here we give a brief description of each construction.

Object extraction from a relative clause is characterised by a relative pronoun (a *wh*-word or *that*) introducing a clause from which an argument in object position has apparently been extracted: *the paper which I wrote*. Our corpus includes cases where the extracted word is (semantically) the object of a preposition in the verb phrase: *the agency that I applied to*.

Object extraction from a reduced relative clause is essentially the same, except that there is no overt relative pronoun: *the paper I wrote; the agency I applied to*. We did not include participial reduced relatives such as *the paper written by the professor*.

Subject extraction from a relative clause is characterised by the apparent extraction of an argument from subject position: *the instrument that measures depth*. A relative pronoun is obligatory in this construction. Our corpus includes passive subjects: *the instrument which was used by the professor*.

Free relatives contain relative pronouns without antecedents: *I heard what she said*, where *what* does not refer to any other noun in the sentence. Free relatives can usually be paraphrased by noun phrases such as *the thing she said* (a standard diagnostic for distinguishing them from embedded interrogatives like *I wonder what she said*). The majority of sentences in our corpus are object free relatives, but we also included some adverbial free relatives: *She told us how to do it*.

Object *wh*-questions are questions in which the *wh*-word is the semantic object of the verb: *What did you eat?*. Objects of prepositions are included: *What city does she live in?*. Also included are a few cases where the *wh*-word is arguably adverbial, but is selected for by the verb: *Where is the park located?*.

Right node raising (RNR) is characterised by coordinated phrases from which a shared element apparently moves to the right: *Mary saw and Susan bought the book*. This construction is unique within our corpus in that the "raised" element can have a wide variety of grammatical functions. Examples include: noun phrase object of verb, noun phrase object of preposition (*material about or messages from the communicator*), a combination of the two (*applied for and won approval*), prepositional phrase modifier (*president and chief executive of the company*), infinitival modifier (*the will and the capacity to prevent the event*), and modified noun (*a good or a bad decision*).

Subject extraction from an embedded clause is characterised by a semantic subject which is ap-

Object extraction from a relative clause

Each must match Wisman's "pie" with the **fragment** that he **carries** with him.

```
dobj(carries, fragment)
```

Object extraction from a reduced relative clause

Put another way, the decline in the yield suggests stocks have gotten pretty rich in price relative to the **dividends** they **pay**, some market analysts say.

```
dobj(pay, dividends)
```

Subject extraction from a relative clause

It consists of a **series** of pipes and a pressure-measuring **chamber** which **record** the rise and fall of the water surface.

```
nsubj(record, series)
nsubj(record, chamber)
```

Free relative

He tried to ignore **what** his own common sense **told** him, but it wasn't possible; her motives were too blatant.

```
dobj(told, what)
```

Object *wh*-question

What **city** does the Tour de France end **in**?

```
pobj(in, city)
```

Right node raising

For the third year in a row, consumers voted Bill Cosby **first** and James Garner **second in** persuasiveness as spokesmen in TV commercials, according to Video Storyboard Tests, New York.

```
prep(first, in)
prep(second, in)
```

Subject extraction from an embedded clause

In assigning to God the **responsibility** which he learned could not **rest** with his doctors, Eisenhower gave evidence of that weakening of the moral intuition which was to characterize his administration in the years to follow.

```
nsubj(rest, responsibility)
```

Table 1: Examples of the seven constructions in the unbounded dependency corpus.

parently extracted across two clause boundaries, as shown in the following bracketing (where $*$ marks the origin of the extracted element): *the responsibility which [the government said [$*$ lay with the voters]]*. Our corpus includes sentences where the embedded clause is a so-called small clause, i.e. one with a null copula verb: *the plan that she considered foolish*, where *plan* is the semantic subject of *foolish.*

2.2 The data

The corpus consists of approximately 100 sentences for each of the seven constructions; 80 of these were reserved for each construction for testing, giving a test set of 560 sentences in total, and the remainder were used for initial experimentation (for example to ensure that default settings for the various parsers were appropriate for this data). We did not annotate the full sentences, since we are only interested in the unbounded dependencies and full annotation of such a corpus would be extremely time-consuming.

With the exception of the question construction, all sentences were taken from the PTB, with roughly half from the WSJ sections (excluding 2-21 which provided the training data for many

of the parsers in our set) and half from Brown (roughly balanced across the different sections). The questions were taken from the question data in Rimell and Clark (2008), which was obtained from various years of the TREC QA track. We chose to use the PTB as the main source because the use of traces in the PTB annotation provides a starting point for the identification of unbounded dependencies.

Sentences were selected for the corpus by a combination of automatic and manual processes. A regular expression applied to PTB trees, searching for appropriate traces for a particular construction, was first used to extract a set of candidate sentences. All candidates were manually reviewed and, if selected, annotated with one or more grammatical relations representing the relevant unbounded dependencies in the sentence. Some of the annotation in the treebank makes identification of some constructions straightforward; for example right node raising is explicitly represented as RNR. Indeed it may have been possible to fully automate this process with use of the `tgrep` search tool. However, in order to obtain reliable statistics regarding frequency of occurrence, and to ensure a high-quality resource, we used fairly broad regular expressions to identify the original set followed by manual review.

We chose to represent the dependencies as grammatical relations (GRs) since this format seemed best suited to represent the kind of semantic relationship we are interested in. GRs are head-based dependencies that have been suggested as a more appropriate representation for general parser evaluation than phrase-structure trees (Carroll et al., 1998). Table 1 gives examples of how GRs are used to represent the relevant dependencies. The particular GR scheme we used was based on the Stanford scheme (de Marneffe et al., 2006); however, the specific GR scheme is not too crucial since the whole sentence is not being represented in the corpus, only the unbounded dependencies.

3 Experiments

The five parsers described in Section 3.2 were used to parse the test sentences in the corpus, and the percentage of dependencies in the test set recovered by each parser for each construction was calculated. The details of how the parsers were run and how the parser output was matched against the gold standard are given in Section 3.3. This is essentially a recall evaluation, and so is open to abuse; for example, a program which returns all the possible word pairs in a sentence, together with all possible labels, would score 100%. However, this is easily guarded against: we simply assume that each parser is being run in a "standard" mode, and that each parser has already been evaluated on a full corpus of GRs in order to measure precision and recall across all dependency types. (Calculating precision for the unbounded dependency evaluation would be difficult since that would require us to know how many *incorrect* unbounded dependencies were returned by each parser.)

Construction	WSJ	Brown	Overall
Obj rel clause	2.3	1.1	1.4
Obj reduced rel	2.7	2.8	2.8
Sbj rel clause	10.1	5.7	7.4
Free rel	2.6	0.9	1.3
RNR	2.2	0.9	1.2
Sbj embedded	2.0	0.3	0.4

Table 2: Frequency of constructions in the PTB (percentage of sentences).

3.1 Statistics relating to the constructions

Table 2 shows the percentage of sentences in the PTB, from those sections that were examined, which contain an example of each type of unbounded dependency. Perhaps not surprisingly, root subject extractions from relative clauses are by far the most common, with the remaining constructions occurring in roughly between 1 and 2% of sentences. Note that, although examples of each individual construction are relatively rare, the combined total is over 10% (assuming that each construction occurs independently). Section 6 contains a discussion regarding the frequency of occurrence of these events and the consequences of this for parser performance.

Table 3 shows the average and maximum distance between head and dependent for each construction, as measured by the difference between word indices. This is a fairly crude measure of distance but gives some indication of how "long-range" the dependencies are for each construction. The cases of object extraction from a relative clause and subject extraction from an embedded clause provide the longest dependencies, on average. The following sentence gives an example of how far apart the head and dependent can be in a

Construction	Avg Dist	Max Dist
Obj rel clause	6.8	21
Obj reduced rel	3.4	8
Sbj rel clause	4.4	18
Free rel	3.4	16
Obj wh-question	4.8	9
RNR	4.8	23
Sbj embedded	7.0	21

Table 3: Distance between head and dependent.

subject embedded construction:

the same **stump** *which had impaled the car of many a guest in the past thirty years and which he refused to have* **removed**.

3.2 The parsers

The parsers that we chose to evaluate are the C&C CCG parser (Clark and Curran, 2007), the Enju HPSG parser (Miyao and Tsujii, 2005), the RASP parser (Briscoe et al., 2006), the Stanford parser (Klein and Manning, 2003), and the DCU post-processor of PTB parsers (Cahill et al., 2004), based on LFG and applied to the output of the Charniak and Johnson reranking parser. Of course we were unable to evaluate every publicly available parser, but we believe these are representative of current wide-coverage robust parsing technology.[1]

The C&C parser is based on CCGbank (Hockenmaier and Steedman, 2007), a CCG version of the Penn Treebank. It is ideally suited for this evaluation because CCG was designed to capture the unbounded dependencies being considered. The Enju parser was designed with a similar motivation to C&C, and is also based on an automatically extracted grammar derived from the PTB, but the grammar formalism is HPSG rather than CCG. Both parsers produce head-word dependencies reflecting the underlying predicate-argument structure of a sentence, and so in theory should be straightforward to evaluate.

The RASP parser is based on a manually constructed POS tag-sequence grammar, with a statistical parse selection component and a robust partial-parsing technique which allows it to return output for sentences which do not obtain a full spanning analysis according to the grammar. RASP has not been designed to capture many of the dependencies in our corpus; for example, the tag-sequence grammar has no explicit representation of verb subcategorisation, and so may not know that there is a missing object in the case of extraction from a relative clause (though it does recover some of these dependencies). However, RASP is a popular parser used in a number of applications, and it returns dependencies in a suitable format for evaluation, and so we considered it to be an appropriate and useful member of our parser set.

The Stanford parser is representative of a large number of PTB parsers, exemplified by Collins (1997) and Charniak (2000). The Parseval scores reported for the Stanford parser are not the highest in the literature, but are competitive enough for our purposes. The advantage of the Stanford parser is that it returns dependencies in a suitable format for our evaluation. The dependencies are obtained by a set of manually defined rules operating over the phrase-structure trees returned by the parser (de Marneffe et al., 2006). Like RASP, the Stanford parser has not been designed to capture unbounded dependencies; in particular it does not make use of any of the trace information in the PTB. However, we wanted to include a "standard" PTB parser in our set to see which of the unbounded dependency constructions it is able to deal with.

Finally, there is a body of work on inserting trace information into the output of PTB parsers (Johnson, 2002; Levy and Manning, 2004), which is the annotation used in the PTB for representing unbounded dependencies. The work which deals with the PTB representation directly, such as Johnson (2002), is difficult for us to evaluate because it does not produce explicit dependencies. However, the DCU post-processor is ideal because it does produce dependencies in a GR format. It has also obtained competitive scores on general GR evaluation corpora (Cahill et al., 2004).

3.3 Parser evaluation

The parsers were run essentially out-of-the-box when parsing the test sentences. The one exception was C&C, which required some minor adjusting of parameters, as described in the parser documentation, to obtain close to full coverage on the data. In addition, the C&C parser comes with a

[1] One obvious omission is any form of dependency parser (McDonald et al., 2005; Nivre and Scholz, 2004). However, the dependencies returned by these parsers are local, and it would be non-trivial to infer from a series of links whether a long-range dependency had been correctly represented. Also, dependency parsers are not significantly better at recovering head-based dependencies than constituent parsers based on the PTB (McDonald et al., 2005).

	Obj RC	Obj Red	Sbj RC	Free	Obj Q	RNR	Sbj Embed	Total
C&C	**59.3**	62.6	80.0	72.6	(**81.2**) 27.5	**49.4**	22.4	(**59.7**) 53.6
Enju	47.3	**65.9**	**82.1**	**76.2**	32.5	47.1	**32.9**	**54.4**
DCU	23.1	41.8	56.8	46.4	27.5	40.8	5.9	35.7
Rasp	16.5	1.1	53.7	17.9	27.5	34.5	15.3	25.3
Stanford	22.0	1.1	74.7	64.3	**41.2**	45.4	10.6	38.1

Table 4: Parser accuracy on the unbounded dependency corpus; the highest score for each construction is in bold; the figures in brackets for C&C derive from the use of a separate question model.

specially designed question model, and so we applied both this and the standard model to the object *wh*-question cases.

The parser output was evaluated against each dependency in the corpus. Due to the various GR schemes used by the parsers, an exact match on the dependency label could not always be expected. We considered a correctly recovered dependency to be one where the gold-standard head and dependent were correctly identified, and the label was an "acceptable match" to the gold-standard label. To be an acceptable match, the label had to indicate the grammatical function of the extracted element at least to the level of distinguishing active subjects, passive subjects, objects, and adjuncts. For example, we allowed an `obj` (object) relation as a close enough match for `dobj` (direct object) in the corpus, even though `obj` does not distinguish different kinds of objects, but we did not allow generic "relative pronoun" relations that are underspecified for the grammatical role of the extracted element.

The differences in GR schemes were such that we ended up performing a time-consuming largely manual evaluation. We list here some of the key differences that made the evaluation difficult.

In some cases, the parser's set of labels was less fine-grained than the gold standard. For example, RASP represents the direct objects of both verbs and prepositions as `dobj` (direct object), whereas the gold-standard uses `pobj` for the preposition case. We counted the RASP output as correctly matching the gold standard.

In other cases, the label on the dependency containing the gold-standard head and dependent was too underspecified to be acceptable by itself. For example, where the gold-standard relation was `dobj(placed,buckets)`, DCU produced `relmod(buckets,placed)` with a generic "relative modifier" label. However, the correct label could be recovered from elsewhere in the parser output, specifically a combination of `relpro(buckets,which)` and `obj(placed,which)`. In this case we counted the DCU output as correctly matching the gold standard.

In some constructions the Stanford scheme, upon which the gold-standard was based, makes different choices about heads than other schemes. For example, in the the phrase *Honolulu, which is the center of the warning system*, the corpus contains a subject dependency with *center* as the head: `nsubj(center,Honolulu)`. Other schemes, however, treat the auxiliary verb *is* as the head of the dependency, rather than the predicate nominal *center*. As long as the difference in head selection was due solely to the idiosyncracies of the GR schemes involved, we counted the relation as correct.

Finally, the different GR schemes treat coordination differently. In the corpus, coordinated elements are always represented with two dependencies. Thus the phrase *they may half* **see** *and half* **imagine** *the old* **splendor** has two gold-standard dependencies: `dobj(see,splendor)` and `dobj(imagine,splendor)`. If a parser produced only the former dependency, but appeared to have the coordination correct, then we awarded two marks, even though the second dependency was not explicitly represented.

4 Results

Accuracies for the various parsers are shown in Table 4, with the highest score for each construction in bold. Enju and C&C are the top performers, operating at roughly the same level of accuracy across most of the constructions. Use of the C&C question model made a huge difference for the *wh*-object construction (81.2% vs. 27.5%), showing that adaptation techniques specific to a particular

construction can be successful (Rimell and Clark, 2008).

In order to learn more from these results, in Section 5 we analyse the various errors made by the C&C parser on each construction. The conclusions that we arrive at for the C&C parser we would also expect to apply to Enju, on the whole, since the design of the two parsers is so similar. In fact, some of the recommendations for improvement on this corpus, such as the need for a better parsing model to make better attachment decisions, are parser independent.

The poor performance of RASP on this corpus is clearly related to a lack of subcategorisation information, since this is crucial for recovering extracted arguments. For Stanford, incorporating the trace information from the PTB into the statistical model in some way is likely to help. The C&C and Enju parsers do this through their respective grammar formalisms. Our informal impression of the DCU post-processor is that it has much of the machinery available to recover the dependencies that the Enju and C&C parsers do, but for some reason which is unclear to us it performs much worse.

5 Analysis of the C&C Parser

We categorised the errors made by the C&C parser on the development data for each construction. We chose the C&C parser for the analysis because it was one of the top performers and we have more knowledge of its workings than those of Enju.

The C&C parser first uses a supertagger to assign a small number of CCG lexical categories (essentially subcategorisation frames) to each word in the sentence. These categories are then combined using a set of combinatory rules to build a CCG derivation. The parser uses a log-linear probability model to select the highest-scoring derivation (Clark and Curran, 2007). In general, errors in dependency recovery may occur if the correct lexical category is not assigned by the supertagger for one or more of the words in a sentence, or if an incorrect derivation is chosen by the parsing model.

For unbounded dependency recovery, one source of errors (labeled **type 1** in Table 5) is the wrong lexical category being assigned to the word (normally a verb or preposition) governing the extraction site. In *these testaments that I would submit here*, if *submit* is assigned a category for an intransitive rather than transitive verb, the verb-object relation will not be recovered.

	1a	1b	1c	1d	2	3	Errs	Tot
ObjRC			6		5	2	13	20
ObjRed	2		1	1	1	3	8	23
SbjRC					8	1	9	43
Free	1					1	2	22
ObjQ			2	2			4	25
RNR			2	1	7	3	13	28
SbjEmb	3	2	1			4	10	13
Subtotal	6	2	12	4				
Total	24				21	14	59	174

Table 5: Error analysis for C&C. *Errs* is the total number of errors for a construction, *Tot* is the number of dependencies of that type in the development data.

There are a number of reasons why the wrong category may be assigned. First, the lexicon may not contain enough information about possible categories for the word (**1a**), or the necessary category may not exist in the parser's grammar at all (**1b**). Even if the grammar contains the correct category and the lexicon makes it available, the parsing model may not choose it (**1c**). Finally, a POS-tagging error on the word may mislead the parser into assigning the wrong category (**1d**).[2]

A second source of errors (**type 2**) is attachment decisions that the parser makes independently of the unbounded dependency. In *Morgan ... carried in several buckets of water from the spring which he poured into the copper boiler*, the parser assigns the correct categories for the relative pronoun and verb, but chooses *spring* rather than *buckets* as the head of the relativized NP (i.e. the object of *pour*). Most attachment errors involve prepositional phrases (PPs) and coordination, which have long been known to be areas where parsers need improvement.

Finally, errors in unbounded dependency recovery may be due to complex errors in the surrounding parse context (**type 3**). We will not comment more on these cases since they do not tell us much about unbounded dependencies in particular.

Table 5 shows the distribution of error types across constructions for the C&C parser. Subject relative clauses, for example, did not have any errors of type 1, because a verb with an extracted

[2]We considered an error to be type 1 only when the category error occurred on the word governing the extraction site, except in the subject embedded sentences, where we also included the embedding verb, since the category of this verb is key to dependency recovery.

subject does not require a special lexical category. Most of the errors here are of type 2. For example, in *a series of pipes and a pressure-measuring chamber which record the rise and fall of the water surface*, the parser attaches the relative clause to *chamber* but not to *series*.

Subject embedded sentences show a different pattern. Many of the errors can be attributed to problems with the lexicon and grammar (1a and 1b). For example, in *shadows that they imagined were Apaches*, the word *imagined* never appears in the training data with the correct category, and so the required entry is missing from the lexicon.

Object extraction from a relative clause had a higher number of errors involving the parsing model (1c). In *the first carefree, dreamless sleep that she had known*, the transitive category is available for *known*, but not selected by the model.

The majority of the errors made by the parser are due to insufficient grammar coverage or weakness in the parsing model due to sparsity of head dependency data, the same fundamental problems that have dogged automatic parsing since its inception. Hence one view of statistical parsing is that it has allowed us to solve the easy problems, but we are still no closer to a general solution for the recovery of the "difficult" dependencies. One possibility is to create more training data targeting these constructions – effectively "active learning by construction" – in the way that Rimell and Clark (2008) were able to build a question parser. We leave this idea for future work.

6 Discussion

Unbounded dependencies are rare events, out in the Zipfian "long tail". They will always constitute a fraction of a percent of the overall total of head-dependencies in any corpus, a proportion too small to make a significant impact on global measures of parser accuracy, when expressive parsers are compared to those that merely approximate human grammar using finite-state or context-free covers. This will remain the case even when such measures are based on dependencies, rather than on parse trees.

Nevertheless, unbounded dependencies remain highly significant in a much more important sense. They support the constructions that are central to those applications of parsing technology for which precision is as important as recall, such as open-domain question-answering. As low-power approximate parsing methods improve (as they must if they are ever to be usable at all for such tasks), we predict that the impact of the constructions we examine here will become evident. No matter how infrequent object questions like "What do frogs eat?" are, if they are answered as if they were subject questions ("Herons"), users will rightly reject any excuse in terms of the overall statistical distribution of related bags of words.

Whether such improvements in parsers come from the availability of more human-labeled data, or from a breakthrough in unsupervised machine learning, we predict an imminent "Uncanny Valley" in parsing applications, due to the inability of parsers to recover certain semantically important dependencies, of the kind familiar from humanoid robotics and photorealistic animation. In such applications, the closer the superficial resemblance to human behavior gets, the more disturbing subtle departures become, and the more they induce mistrust and revulsion in the user.

7 Conclusion

In this paper we have demonstrated that current parsing technology is poor at recovering some of the unbounded dependencies which are crucial for fully representing the underlying predicate-argument structure of a sentence. We have also argued that correct recovery of such dependencies will become more important as parsing technology improves, despite the relatively low frequency of occurrence of the corresponding grammatical constructions. We also see this more focused parser evaluation methodology — in this case construction-focused — as a way of improving parsing technology, as an alternative to the exclusive focus on incremental improvements in overall accuracy measures such as Parseval.

Acknowledgments

Laura Rimell and Stephen Clark were supported by EPSRC grant EP/E035698/1. Mark Steedman was supported by EU IST Cognitive Systems grant IP FP6-2004-IST-4-27657 (PACO-PLUS). We would like to thank Aoife Cahill for producing the DCU data.

References

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *HLT '91: Proceedings of the Workshop on Speech and Natural Language*, pages 306–311.

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings of the 10th Meeting of the EACL*, pages 19–26, Budapest, Hungary.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the Interactive Demo Session of COLING/ACL-06*, Sydney, Australia.

A. Cahill, M. Burke, R. O'Donovan, J. van Genabith, and A. Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of the 42nd Meeting of the ACL*, pages 320–327, Barcelona, Spain.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. Dynamic programming and the perceptron for efficient, feature-rich parsing. In *Proceedings of the Twelfth Conference on Natural Language Learning (CoNLL-08)*, pages 9–16, Manchester, UK.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st LREC Conference*, pages 447–454, Granada, Spain.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Meeting of the ACL*, pages 173–180, Michigan, Ann Arbor.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the NAACL*, pages 132–139, Seattle, WA.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Stephen Clark, Mark Steedman, and James R. Curran. 2004. Object-extraction and question-parsing using CCG. In *Proceedings of the EMNLP Conference*, pages 111–118, Barcelona, Spain.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Meeting of the ACL*, pages 16–23, Madrid, Spain.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th LREC Conference*, Genoa, Italy.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 EMNLP Conference*, Pittsburgh, PA.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the 46th Meeting of the ACL*, pages 586–594, Columbus, Ohio.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Meeting of the ACL*, pages 136–143, Philadelphia, PA.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

Roger Levy and Christopher Manning. 2004. Deep dependencies from context-free statistical parsers: correcting the surface dependency approximation. In *Proceedings of the 42nd Meeting of the ACL*, pages 328–335, Barcelona, Spain.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of IJCAI-95*, pages 1420–1425, Montreal, Canada.

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

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Meeting of the ACL*, pages 91–98, Michigan, Ann Arbor.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd Meeting of the ACL*, pages 83–90, Michigan, Ann Arbor.

J. Nivre and M. Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING-04*, pages 64–70, Geneva, Switzerland.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of the HLT/NAACL Conference*, Rochester, NY.

Laura Rimell and Stephen Clark. 2008. Adapting a lexicalized-grammar parser to contrasting domains. In *Proceedings of the 2008 EMNLP Conference*, pages 475–484, Honolulu, Hawai'i.

Parser Adaptation and Projection with Quasi-Synchronous Grammar Features*

David A. Smith
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003, USA
dasmith@cs.umass.edu

Jason Eisner
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218, USA
jason@cs.jhu.edu

Abstract

We connect two scenarios in structured learning: **adapting** a parser trained on one corpus to another annotation style, and **projecting** syntactic annotations from one language to another. We propose **quasi-synchronous grammar** (QG) features for these structured learning tasks. That is, we score a aligned pair of source and target trees based on local features of the trees and the alignment. Our quasi-synchronous model assigns positive probability to any alignment of any trees, in contrast to a synchronous grammar, which would insist on some form of structural parallelism.

In monolingual dependency parser adaptation, we achieve high accuracy in translating among multiple annotation styles for the same sentence. On the more difficult problem of cross-lingual parser projection, we learn a dependency parser for a target language by using bilingual text, an English parser, and automatic word alignments. Our experiments show that unsupervised QG projection improves on parses trained using only high-precision projected annotations and far outperforms, by more than 35% absolute dependency accuracy, learning an unsupervised parser from raw target-language text alone. When a few target-language parse trees are available, projection gives a boost equivalent to doubling the number of target-language trees.

*The first author would like to thank the Center for Intelligent Information Retrieval at UMass Amherst. We would also like to thank Noah Smith and Rebecca Hwa for helpful discussions and the anonymous reviewers for their suggestions for improving the paper.

1 Introduction

1.1 Parser Adaptation

Consider the problem of learning a dependency parser, which must produce a directed tree whose vertices are the words of a given sentence. There are many differing conventions for representing syntactic relations in dependency trees. Say that we wish to output parses in the Prague style and so have annotated a small **target corpus**—e.g., 100 sentences—with those conventions. A parser trained on those hundred sentences will achieve mediocre dependency accuracy (the proportion of words that attach to their correct parent).

But what if we also had a large number of trees in the CoNLL style (the **source corpus**)? Ideally they should help train our parser. But unfortunately, a parser that learned to produce perfect CoNLL-style trees would, for example, get both links "wrong" when its coordination constructions were evaluated against a Prague-style gold standard (Figure 1).

If it were just a matter of this one construction, the obvious solution would be to write a few rules by hand to transform the large source training corpus into the target style. Suppose, however, that there were many more ways that our corpora differed. Then we would like to *learn a statistical model to transform one style of tree into another.*

We may not possess hand-annotated training data for this tree-to-tree transformation task. That would require the two corpora to annotate some of the *same* sentences in different styles.

But fortunately, we can automatically obtain a noisy form of the necessary paired-tree training data. A parser trained on the source corpus can parse the sentences in our target corpus, yielding trees (or more generally, probability distributions over trees) in the source style. We will then learn a tree transformation model relating these noisy source trees to our known trees in the target style.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 822–831,
Singapore, 6-7 August 2009.

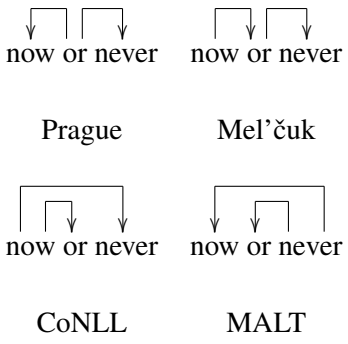


Figure 1: Four of the five logically possible schemes for annotating coordination show up in human-produced dependency treebanks. (The other possibility is a reverse Mel'čuk scheme.) These treebanks also differ on other conventions.

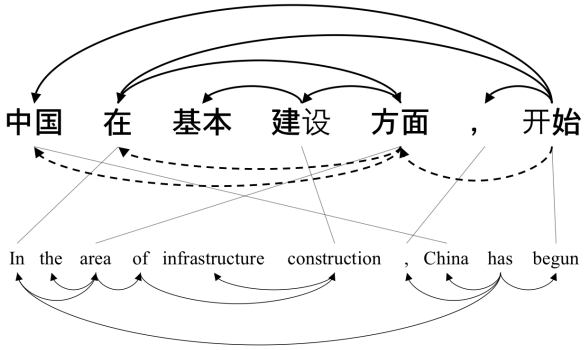


Figure 2: With the English tree and alignment provided by a parser and aligner at test time, the Chinese parser finds the correct dependencies (see §6). A monolingual parser's incorrect edges are shown with dashed lines.

This model should enable us to convert the original large source corpus to target style, giving us additional training data in the target style.

1.2 Parser Projection

For many target languages, however, we do not have the luxury of a large parsed "source corpus" in the language, even one in a different style or domain as above. Thus, we may seek other forms of data to augment our small target corpus. One option would be to leverage unannotated text (McClosky et al., 2006; Smith and Eisner, 2007). But we can also try to transfer syntactic information from a parsed source corpus *in another language*. This is an extreme case of out-of-domain data. This leads to the second task of this paper: *learning a statistical model to transform a syntactic analysis of a sentence in one language into an analysis of its translation*.

Tree transformations are often modeled with *synchronous* grammars. Suppose we are given a sentence w' in the "source" language and its translation w into the "target" language. Their syntactic parses t' and t are presumably not independent, but will tend to have some parallel or at least correlated structure. So we could *jointly* model the parses t', t and the alignment a between them, with a model of the form $p(t, a, t' \mid w, w')$.

Such a joint model captures how t, a, t' mutually constrain each other, so that even partial knowledge of some of these three variables can help us to recover the others when training or decoding on bilingual text. This idea underlies a number of recent papers on syntax-based alignment (using t and t' to better recover a), grammar induction from bitext (using a to better recover t and t'), parser projection (using t' and a to better recover t), as well as full joint parsing (Smith and Smith, 2004; Burkett and Klein, 2008).

In this paper, we condition on the 1-best source tree t'. As for the alignment a, our models either condition on the 1-best alignment or integrate the alignment out. Our models are thus of the form $p(t \mid w, w', t', a)$ or, in the generative case, $p(w, t, a \mid w', t')$. We intend to consider other formulations in future work.

So far, this is very similar to the monolingual parser adaptation scenario, but there are a few key differences. Since the source and target sentences in the bitext are in different languages, there is no longer a trivial alignment between the words of the source and target trees. Given word alignments, we could simply try to project dependency links in the source tree onto the target text. A link-by-link projection, however, could result in invalid trees on the target side, with cycles or disconnected words. Instead, our models learn the necessary transformations that align and transform a source tree into a target tree by means of quasi-synchronous grammar (QG) features.

Figure 2 shows an example of bitext helping disambiguation when a parser is trained with only a small number of Chinese trees. With the help of the English tree and alignment, the parser is able to recover the correct Chinese dependencies using QG features. Incorrect edges from the monolingual parser are shown with dashed lines. (The bilingual parser corrects additional errors in the second half of this sentence, which has been removed to improve legibility.) The parser is able to recover the long-distance dependency from the first Chinese word (*China*) to the last (*begun*), while skipping over the intervening noun

phrase that confused the undertrained monolingual parser. Although, due to the auxiliary verb, "China" and "begun" are *siblings* in English and not in direct dependency, the QG features still leverage this indirect projection.

1.3 Plan of the Paper

We start by describing the features we use to augment conditional and generative parsers when scoring pairs of trees (§2). Then we discuss in turn monolingual (§3) and cross-lingual (§4) parser adaptation. Finally, we present experiments on cross-lingual parser projection in conditions when no target language trees are available for training (§5) and when some trees are available (§6).

2 Form of the Model

What should our model of source and target trees look like? In our view, traditional approaches based on synchronous grammar are problematic both computationally and linguistically. Full inference takes $O(n^6)$ time or worse (depending on the grammar formalism). Yet synchronous models only consider a limited hypothesis space: e.g., parses must be projective, and alignments must decompose according to the recursive parse structure. (For example, two nodes can be aligned only if their respective parents are also aligned.) The synchronous model's probability mass function is also restricted to decompose in this way, so it makes certain conditional independence assumptions; put another way, it can evaluate only certain properties of the triple (t, a, t').

We instead model (t, a, t') as an *arbitrary graph* that includes dependency links among the words of each sentence as well as arbitrary alignment links between the words of the two sentences. This permits non-synchronous and many-to-many alignments. The only hard constraint we impose is that the dependency links within each sentence must constitute a valid monolingual parse—a directed projective spanning tree.[1]

Given the two sentences w, w', our probability distribution over possible graphs considers local features of the parses, the alignment, and both jointly. Thus, we learn what local syntactic configurations tend to occur in each language and how they correspond across languages. As a result, we might learn that parses are "mostly synchronous," but that there are some systematic cross-linguistic divergences and some instances of sloppy (non-parallel or inexact) translation. Our model is thus a form of quasi-synchronous grammar (QG) (Smith and Eisner, 2006a). In that paper, QG was applied to word alignment and has since found applications in question answering (Wang et al., 2007), paraphrase detection (Das and Smith, 2009), and machine translation (Gimpel and Smith, 2009).

All the models in this paper are conditioned on the source tree t'. Conditionally-trained models of adaptation and projection also condition on the target string w and its alignment a to w' and thus have the form $p(t \mid w, w', t', a)$; the unsupervised, generative projection models in §5 have the form $p(w, t, a \mid w', t')$.

The score s of a given tuple of trees, words, and alignment can thus be written as a dot product of weights $\mathbf{w}$ with features $\mathbf{f}$ and $\mathbf{g}$:

$$s(t, t', a, w, w') = \sum_i w_i f_i(t, w) + \sum_j w_j g_j(t, t', a, w, w')$$

The features $\mathbf{f}$ look only at target words and dependencies. In the conditional models of §3 and §6, these features are those of an edge-factored dependency parser (McDonald et al., 2005). In the generative models of §5, $\mathbf{f}$ has the form of a dependency model with valence (Klein and Manning, 2004). All models, for instance, have a feature template that considers the parts of speech of a potential parent-child relation.

In order to benefit from the source language, we also need to include bilingual features $\mathbf{g}$. When scoring a candidate target dependency link from word $x \rightarrow y$, these features consider the relationship of their corresponding source words x' and y'. (The correspondences are determined by the alignment a.) For instance, the source tree t' may contain the link $x' \rightarrow y'$, which would cause a feature for *monotonic* projection to fire for the $x \rightarrow y$ edge. If, on the other hand, $y' \rightarrow x' \in t'$, a *head-swapping* feature fires. If $x' = y'$, i.e. x and y align to the same word, the *same-word* feature fires. Similar features fire when x' and y' are in grandparent-grandchild, sibling, c-command, or none-of-the above relationships, or when y aligns to NULL. These alignment classes are called *configurations* (Smith and Eisner, 2006a, and following). When training is conditioned on the target words (see §3 and §6 below), we conjoin these

[1] Non-projective parsing would also be possible.

configuration features with the part of speech and coarse part of speech of one or both of the source and target words, i.e. the feature template has from one to four tags.

In conditional training, the exponentiated scores s are normalized by a constant: $Z = \sum_t \exp[s(t, t', a, w, w')]$. For the generative model, the locally normalized generative process is explained in §5.3.4.

Previous researchers have written fix-up rules to massage the projected links after the fact and learned a parser from the resulting trees (Hwa et al., 2005). Instead, our models learn the necessary transformations that align and transform a source tree into a target tree. Other researchers have tackled the interesting task of learning parsers from unparsed bitext alone (Kuhn, 2004; Snyder et al., 2009); our methods take advantage of investments in high-resource languages such as English. In work most closely related to this paper, Ganchev et al. (2009) constrain the posterior distribution over target-language dependencies to align to source dependencies some "reasonable" proportion of the time ($\approx 70\%$, cf. Table 2 in this paper). This approach performs well but cannot directly learn regular cross-language non-isomorphisms; for instance, some fixup rules for auxiliary verbs need to be introduced. Finally, Huang et al. (2009) use features, somewhat like QG configurations, on the shift-reduce actions in a monolingual, target-language parser.

3 Adaptation

As discussed in §1, the adaptation scenario is a special case of parser projection where the word alignments are one-to-one and observed. To test our handling of QG features, we performed experiments in which training saw the correct parse trees in both source and target domains, and the mapping between them was simple and regular. We also performed experiments where the source trees were replaced by the noisy output of a trained parser, making the mapping more complex and harder to learn.

We used the subset of the Penn Treebank from the CoNLL 2007 shared task and converted it to dependency representation while varying two parameters: (1) CoNLL vs. Prague coordination style (Figure 1), and (2) preposition the head vs. the child of its nominal object.

We trained an edge-factored dependency parser (McDonald et al., 2005) on "source" domain data that followed one set of dependency conventions. We then trained an edge-factored parser with QG features on a small amount of "target" domain data. The source parser outputs were produced for all target data, both training and test, so that features for the target parser could refer to them.

In this task, we know what the gold-standard source language parses are for any given text, since we can produce them from the original Penn Treebank. We can thus measure the contribution of adaptation loss alone, and the combined loss of imperfect source-domain parsing with adaptation (Table 1). When no target domain trees are available, we simply have the performance of the source domain parser on this out-of-domain data. Training a target-domain parser on as few as 10 sentences shows substantial improvements in accuracy. In the "gold" conditions, where the target parser starts with perfect source trees, accuracy approaches 100%; in the realistic "parse" conditions, where the target-domain parser gets noisy source-domain parses, the improvements are quite significant but approach a lower ceiling imposed by the performance of the source parser.[2]

The adaptation problem in this section is a simple proof of concept of the QG approach; however, more complex and realistic adaptation problems exist. Monolingual adaptation is perhaps most obviously useful when the source parser is a black-box or rule-based system or is trained on unavailable data. One might still want to use such a parser in some new context, which might require new data or a new annotation standard.

We are also interested in scenarios where we want to avoid expensive retraining on large reannotated treebanks. We would like a linguist to be able to annotate a few trees according to a hypothesized theory and then quickly use QG adaptation to get a parser for that theory. One example would be adapting a constituency parser to produce dependency parses. We have concentrated here on adapting between two dependency parse styles, in order to line up with the cross-lingual tasks to which we now turn.

[2] In the diagonal cells, source and target styles match, so training the QG parser amounts to a "stacking" technique (Martins et al., 2008). The small training size and overregularization of the QG parser mildly hurts in-domain parsing performance.

	% Dependency Accuracy on Target											
	CoNLL-PrepHead			CoNLL-PrepChild			Prague-PrepHead			Prague-PrepChild		
Source	0	10	100	0	10	100	0	10	100	0	10	100
Gold CoNLL-PrepHead	100	99.6	99.6	79.5	96.9	97.8	90.5	95.0	98.1	71.0	92.7	95.4
Parse CoNLL-PrepHead	89.5	88.9	89.0	71.4	85.9	87.9	82.5	84.3	87.8	65.2	82.2	86.1
Gold CoNLL-PrepChild	79.5	96.6	97.3	100	99.6	99.6	71.0	91.3	95.5	89.9	94.5	97.9
Parse CoNLL-PrepChild	71.0	84.2	86.8	88.1	87.5	88.0	64.9	80.7	84.9	80.9	83.5	86.1
Gold Prague-PrepHead	90.5	95.5	96.7	71.0	92.0	94.2	100	99.6	99.6	79.6	97.4	98.1
Parse Prague-PrepHead	83.0	87.1	87.4	65.6	84.2	85.9	88.5	88.3	88.0	70.7	86.4	86.8
Gold Prague-PrepChild	71.0	91.6	93.8	89.9	95.6	96.4	79.6	96.0	97.1	100	99.6	99.6
Parse Prague-PrepChild	65.3	81.7	84.6	81.2	84.5	86.1	70.4	83.2	85.3	86.9	86.1	86.8

Table 1: Adapting a parser to a new annotation style. We learn to parse in a "target" style (wide column label) given some number (narrow column label) of supervised target-style training sentences. As a font of additional features, *all training and test sentences* have already been augmented with parses in some "source" style (row label): either gold-standard parses (an oracle experiment) or else the output of a parser trained on 18k source trees (more realistic). If we have 0 training sentences, we simply output the source-style parse. But with 10 or 100 target-style training sentences, each off-diagonal block learns to adapt, mostly closing the gap with the diagonal block in the same column. In the diagonal blocks, source and target styles match, and the QG parser degrades performance when acting as a "stacked" parser.

4 Cross-Lingual Projection: Background

As in the adaptation scenario above, many syntactic structures can be transferred from one language to another. In this section, we evaluate the extent of this direct projection on a small hand-annotated corpus. In §5, we will use a QG generative model to learn dependency parsers from bitext when there are no annotations in the target language. Finally, in §6,we show how QG features can augment a target-language parser trained on a small set of labeled trees.

For syntactic annotation projection to work at all, we must hypothesize, or observe, that at least some syntactic structures are preserved in translation. Hwa et al. (2005) have called this intuition the **Direct Correspondence Assumption** (DCA, with slight notational changes):

> Given a pair of sentences w and w' that are translations of each other with syntactic structure t and t', if nodes x' and y' of t' are aligned with nodes x and y of t, respectively, and if syntactic relationship $R(x', y')$ holds in t', then $R(x, y)$ holds in t.

The validity of this assumption clearly depends on the node-to-node alignment of the two trees. We again work in a dependency framework, where syntactic nodes are simply lexical items. This allows us to use existing work on word alignment.

Hwa et al. (2005) tested the DCA under idealized conditions by obtaining hand-corrected dependency parse trees of a few hundred sentences of Spanish-English and Chinese-English bitext. They also used human-produced word alignments.

Corpus	Prec.[%]	Rec.[%]
Spanish	64.3	28.4
(no punc.)	72.0	30.8
Chinese	65.1	11.1
(no punc.)	68.2	11.5

Table 2: Precision and recall of direct dependency projection via one-to-one links alone.

Since their word alignments could be many-to-many, they gave a heuristic Direct Projection Algorithm (DPA) for resolving them into component dependency relations. It should be noted that this process introduced empty words into the projected target language tree and left words that are unaligned to English detached from the tree; as a result, they measured performance in dependency F-score rather than accuracy. With manual English parses and word alignments, this DPA achieved 36.8% F-score in Spanish and 38.1% in Chinese. With Collins-model English parses and GIZA++ word alignments, F-score was 33.9% for Spanish and 26.3% for Chinese. Compare this to the Spanish attach-left baseline of 31.0% and the Chinese attach-right baselines of 35.9%. These discouragingly low numbers led them to write language-specific transformation rules to fix up the projected trees. After these rules were applied to the projections of automatic English parses, F-score was 65.7% for English and 52.4% for Chinese.

While these F-scores were low, it is useful to look at a subset of the alignment: dependencies projected across one-to-one alignments before the heuristic fix-ups had a much higher precision, if lower recall, than Hwa et al.'s final results. Us-

ing Hwa et al.'s data, we calculated that the precision of projection to Spanish and Chinese via these one-to-one links was $\approx 65\%$ (Table 2). There is clearly more information in these direct links than one would think from the F-scores. To exploit this information, however, we need to overcome the problems of (1) learning from partial trees, when not all target words are attached, and (2) learning in the presence of the still considerable noise in the projected one-to-one dependencies—e.g., at least 28% error for Spanish non-punctuation dependencies.

What does this noise consist of? Some errors reflect fairly arbitrary annotation conventions in treebanks, e.g. should the auxiliary verb govern the main verb or vice versa. (Examples like this suggest that the projection problem contains the adaptation problem above.) Other errors arise from divergences in the complements required of certain head words. In the German-English translation pair, with co-indexed words aligned,

$$[\text{an } [\text{den Libanon}_1]]\ \text{denken}_2 \ \leftrightarrow\ \text{remember}_2\ \text{Lebanon}_1$$

we would prefer that the preposition *an* attach to *denken*, even though the preposition's object *Libanon* aligns to a direct child of *remember*. In other words, we would like the grandparent-parent-child chain of *denken* $\rightarrow$ *an* $\rightarrow$ *Libanon* to align to the parent-child pair of *remember* $\rightarrow$ *Lebanon*. Finally, naturally occurring bitexts contain some number of free or erroneous translations. Machine translation researchers often seek to strike these examples from their training corpora; "free" translations are not usually welcome from an MT system.

5 Unsupervised Cross-Lingual Projection

First, we consider the problem of parser projection when there are zero target-language trees available. As in much other work on unsupervised parsing, we try to learn a generative model that can predict target-language sentences. Our novel contribution is to *condition the probabilities of the generative actions* on the dependency parse of a source-language translation. Thus, our generative model is a quasi-synchronous grammar, exactly as in (Smith and Eisner, 2006a).[3]

When training on target sentences w, therefore, we tune the model parameters to maximize not $\sum_t p(t, w)$ as in ordinary EM, but rather $\sum_t p(t, w, a \mid t', w')$. We hope that this *conditional EM* training will drive the model to posit appropriate syntactic relationships in the latent variable t, because—thanks to the structure of the QG model—that is the easiest way for it to exploit the extra information in t', w' to help predict w.[4] At test time, t', w' are not made available, so we just use the trained model to find $\text{argmax}_t\, p(t \mid w)$, backing off from the conditioning on t', w' and summing over a.

Below, we present the specific generative model (§5.1) and some details of training (§5.2). We will then compare three approaches (§5.3):

§**5.3.2** a straight EM baseline (which does not condition on t', w' at all)

§**5.3.3** a "hard" projection baseline (which naively projects t', w' to derive direct supervision in the target language)

§**5.3.4** our conditional EM approach above (which makes t', w' available to the learner for "soft" indirect supervision via QG)

5.1 Generative Models

Our base models of target-language syntax are generative dependency models that have achieved state-of-the art results in unsupervised dependency structure induction. The simplest version, called Dependency Model with Valence (DMV), has been used in isolation and in combination with other models (Klein and Manning, 2004; Smith and Eisner, 2006b). The DMV generates the right children, and then independently the left children, for each node in the dependency tree. Nodes correspond to words, which are represented by their part-of-speech tags. At each step of generation, the DMV stochastically chooses whether to stop generating, conditioned on the currently generating head; whether it is generating to the right or left; and whether it has yet generated any children on that side. If it chooses to continue, it then

[3] Our task here is new; they used it for alignment.

[4] The contrastive estimation of Smith and Eisner (2005) also used a form of conditional EM, with similar motivation. They suggested that EM grammar induction, which learns to predict w, unfortunately learns mostly to predict lexical topic or other properties of the training sentences that do not strongly require syntactic latent variables. To focus EM on modeling the *syntactic* relationships, they conditioned the prediction of w on almost complete knowledge of the lexical items. Similarly, we condition on a source translation of w. Furthermore, our QG model structure makes it easy for EM to learn to exploit the (explicitly represented) syntactic properties of that translation when predicting w.

stochastically generates the tag of a new child, conditioned on the head. The parameters of the model are thus of the form

$$p(stop \mid head, dir, adj) \quad (1)$$

$$p(child \mid head, dir) \quad (2)$$

where $head$ and $child$ are part-of-speech tags, $dir \in \{\text{left}, \text{right}\}$, and $adj, stop \in \{\text{true}, \text{false}\}$. ROOT is stipulated to generate a single right child.

Bilingual configurations that condition on t', w' (§2) are incorporated into the generative process as in Smith and Eisner (2006a). When the model is generating a new child for word x, aligned to x', it first chooses a configuration and then chooses a source word y' in that configuration. The child y is then generated, conditioned on its parent x, most recent sibling a, and its source analogue y'.

5.2 Details of EM Training

As in previous work on grammar induction, we learn the DMV from part-of-speech-tagged target-language text. We use expectation maximization (EM) to maximize the likelihood of the data. Since the likelihood function is nonconvex in the unsupervised case, our choice of initial parameters can have a significant effect on the outcome. Although we could also try many random starting points, the initializer in Klein and Manning (2004) performs quite well.

The base dependency parser generates the right dependents of a head separately from the left dependents, which allows $O(n^3)$ dynamic programming for an n-word target sentence. Since the QG annotates nonterminals of the grammar with single nodes of t', and we consider two nodes of t' when evaluating the above dependency configurations, QG parsing runs in $O(n^3m^2)$ for an m-word source sentence. If, however, we restrict candidate senses for a target child c to come from links in an IBM Model 4 Viterbi alignment, we achieve $O(n^3k^2)$, where k is the maximum number of possible words aligned to a given target language word. In practice, $k \ll m$, and parsing is not appreciably slower than in the monolingual setting.

If all configurations were equiprobable, the source sentence would provide no information to the target. In our QG experiments, therefore, we started with a bias towards direct parent–child links and a very small probability for breakages of locality. The values of other configuration parameters seem, experimentally, less important for insuring accurate learning.

5.3 Experiments

Our experiments compare learning on target language text to learning on parallel text. In the latter case, we compare learning from high-precision one-to-one alignments alone, to learning from all alignments using a QG.

5.3.1 Corpora

Our development and test data were drawn from the German TIGER and Spanish Cast3LB treebanks as converted to projective dependencies for the CoNLL 2007 Shared Task (Brants et al., 2002; Civit Torruella and Martí Antonín, 2002).[5]

Our training data were subsets of the 2006 Statistical Machine Translation Workshop Shared Task, in particular from the German-English and Spanish-English Europarl parallel corpora (Koehn, 2002). The Shared Task provided pre-built automatic GIZA++ word alignments, which we used to facilitate replicability. Since these word alignments do not contain posterior probabilities or null links, nor do they distinguish which links are in the IBM Model intersection, we treated all links as equally likely when learning the QG. Target language words unaligned to any source language words were the only nodes allowed to align to NULL in QG derivations.

We parsed the English side of the bitext with the projective dependency parser described by McDonald et al. (2005) trained on the Penn Treebank §§2–20. Much previous work on unsupervised grammar induction has used gold-standard part-of-speech tags (Smith and Eisner, 2006b; Klein and Manning, 2004; Klein and Manning, 2002). While there are no gold-standard tags for the Europarl bitext, we did train a conditional Markov

[5] We made one change to the annotation conventions in German: in the dependencies provided, words in a noun phrase governed by a preposition were all attached to that preposition. This meant that in the phrase *das Kind* ("the child") in, say, subject position, *das* was the child of *Kind*; but, in *für das Kind* ("for the child"), *das* was the child of *für*. This seems to be a strange choice in converting from the TIGER constituency format, which does in fact annotate NPs inside PPs; we have standardized prepositions to govern only the head of the noun phrase. We did *not* change any other annotation conventions to make them more like English. In the Spanish treebank, for instance, control verbs are the children of their verbal complements: in *quiero decir* ("I want to say"="I mean"), *quiero* is the child of *decir*. In German coordinations, the coordinands all attach to the first, but in English, they all attach to the last. These particular divergences in annotation style hurt all of our models equally (since none of them have access to labeled trees). These annotation divergences are one motivation for experiments below that include some target trees.

Baselines	Dependency accuracy [%] German	Spanish
Modify prev.	18.2	28.5
Modify next	27.5	21.4
EM	30.2	25.6
Hard proj.	66.2	59.1
Hard proj. w/EM	58.6	53.0
QG w/EM	**68.5**	**64.8**

Table 3: Test accuracy with unsupervised training methods

model tagger on a few thousand tagged sentences. This is the only supervised data we used in the target. We created versions of each training corpus with the first thousand, ten thousand, and hundred thousand sentence pairs, each a prefix of the next. Since the target-language-only baseline converged much more slowly, we used a version of the corpora with sentences 15 target words or fewer.

5.3.2 Fully Unsupervised EM

Using the target side of the bitext as training data, we initialized our model parameters as described in §5.2 and ran EM. We checked convergence on a development set and measured unlabeled dependency accuracy on held-out test data. We compare performance to simple attach-right and attach left baselines (Table 3). For mostly head-final German, the "modify next" baseline is better; for mostly head-initial Spanish, "modify previous" wins. Even after several hundred iterations, performance was slightly, but not significantly better than the baseline for German. EM training did not beat the baseline for Spanish.[6]

5.3.3 Hard Projection, Semi-Supervised EM

The simplest approach to using the high-precision one-to-one word alignments is labeled "hard projection" in the table. We filtered the training corpus to find sentences where enough links were projected to completely determine a target language tree. Of course, we needed to filter more than 1000 sentences of bitext to output 1000 training sentences in this way. We simply perform supervised training with this subset, which is still quite noisy (§4), and performance quickly plateaus. Still, this method substantially improves over the baselines and unsupervised EM.

Restricting ourselves to fully projected trees seems a waste of information. We can also simply take all one-to-one projected links, impute expected counts for the remaining dependencies with EM, and update our models. This approach ("hard projection with EM"), however, performed worse than using only the fully projected trees. In fact, only the first iteration of EM with this method made any improvement; afterwards, EM degraded accuracy further from the numbers in Table 3.

5.3.4 Soft Projection: QG & Conditional EM

The quasi-synchronous model used all of the alignments in re-estimating its parameters and performed significantly better than hard projection. Unlike EM on the target language alone, the QG's performance does not depend on a clever initializer for initial model weights—all parameters of the generative model except for the QG configuration features were initialized to zero. Setting the prior to prefer direct correspondence provides the necessary bias to initialize learning.

Error analysis showed that certain types of dependencies eluded the QG's ability to learn from bitext. The Spanish treebank treats some verbal complements as the heads of main verbs and auxiliary verbs as the children of participles; the QG, following the English, learned the opposite dependency direction. Spanish treebank conventions for punctuation were also a common source of errors. In both German and Spanish, coordinations (a common bugbear for dependency grammars) were often mishandled: both treebanks attach the later coordinands and any conjunctions to the first coordinand; the reverse is true in English. Finally, in both German and Spanish, preposition attachments often led to errors, which is not surprising given the unlexicalized target-language grammars. Rather than trying to adjudicate which dependencies are "mere" annotation conventions, it would be useful to test learned dependency models on some extrinsic task such as relation extraction or machine translation.

6 Supervised Cross-Lingual Projection

Finally, we consider the problem of parser projection when some target language trees are available. As in the adaptation case (§3), we train a conditional model (*not* a generative DMV) of the target

[6] While these results are worse than those obtained previously for this model, the experiments in Klein and Manning (2004) and only used sentences of 10 words or fewer, without punctuation, and with gold-standard tags. Punctuation in particular seems to trip up the initializer: since a sentence-final periods appear in most sentences, EM often decides to make it the head.

tree given the target sentence, using the monolingual and bilingual QG features, including configurations conjoined with tags, outlined above (§2).

For these experiments, we used the LDC's English-Chinese Parallel Treebank (ECTB). Since manual word alignments also exist for a part of this corpus, we were able to measure the loss in accuracy (if any) from the use of an automatic English parser and word aligner. The source-language English dependency parser was trained on the Wall Street Journal, where it achieved 91% dependency accuracy on development data. However, it was only 80.3% accurate when applied to our task, the English side of the ECTB.[7]

After parsing the source side of the bitext, we train a parser on the annotated target side, using QG features described above (§2). Both the monolingual target-language parser and the projected parsers are trained to optimize conditional likelihood of the target trees t' with ten iterations of stochastic gradient ascent.

In Figure 3, we plot the performance of the target-language parser on held-out bitext. Although projection performance is, not surprisingly, better if we know the true source trees at training and test time, even with the 1-best output of the source parser, QG features help produce a parser as accurate asq one trained on twice the amount of monolingual data. In ablation experiments, we included bilingual features only for directly projected links, with no features for head-swapping, grandparents, etc. When using 1-best English parses, parsers trained only with direct-projection and monolingual features performed worse; when using gold English parses, parsers with direct-projection-only features performed better when trained with more Chinese trees.

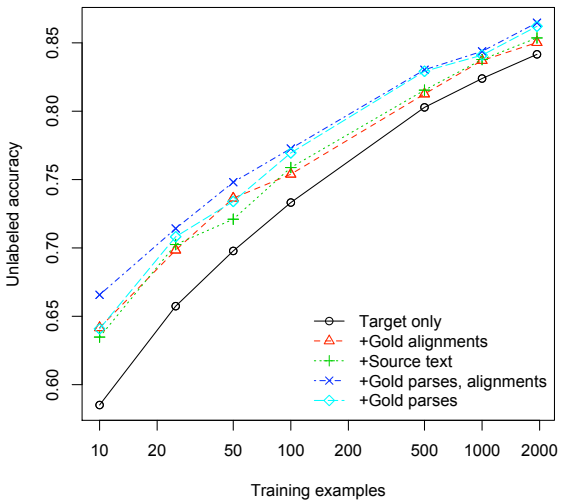


Figure 3: Parser projection with target trees. Using the true or 1-best parse trees in the source language is equivalent to having twice as much data in the target language. Note that the penalty for using automatic alignments instead of gold alignments is negligible; in fact, using *Source text* alone is often higher than +*Gold alignments*. Using gold source trees, however, significantly outperforms using 1-best source trees.

7 Discussion

The two related problems of parser adaptation and projection are often approached in different ways. Many adaptation methods operate by simple augmentations of the target feature space, as we have done here (Daume III, 2007). Parser projection, on the other hand, often uses a multi-stage pipeline (Hwa et al., 2005). The methods presented here move parser projection much closer in efficiency and simplicity to monolingual parsing.

We showed that augmenting a target parser with quasi-synchronous features can lead to significant improvements—first in experiments with adapting to different dependency representations in English, and then in cross-language parser projection. As with many domain adaptation problems, it is quite helpful to have some annotated target data, especially when annotation styles vary (Dredze et al., 2007). Our experiments show that unsupervised QG projection improves on parsers trained using only high-precision projected annotations and far outperforms, by more than 35% absolute dependency accuracy, unsupervised EM. When a small number of target-language parse trees is available, projection gives a boost equivalent to doubling the number of target trees.

The loss in performance from conditioning only on noisy 1-best source parses points to some natural avenues for improvement. We are exploring methods that incorporate a packed parse forest on the source side and similar representations of uncertainty about alignments. Building on our recent belief propagation work (Smith and Eisner, 2008), we can jointly infer two dependency trees and their alignment, under a joint distribution $p(t, a, t' \mid w, w')$ that evaluates the full graph of dependency and alignment edges.

[7] It would be useful to explore whether the techniques of §3 above could be used to improve English accuracy by domain adaptation. In theory a model with QG features trained to perform well on Chinese should not suffer from an inaccurate, but consistent, English parser, but the results in Figure 3 indicate a significant benefit to be had from better English parsing or from joint Chinese-English inference.

References

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *TLT*.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *EMNLP*.

M. Civit Torruella and M. A. Martí Antonín. 2002. Design principles for a Spanish treebank. In *TLT*.

Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *ACL-IJCNLP*.

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

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, João Graca, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1051–1055.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *ACL-IJCNLP*.

Kevin Gimpel and Noah A. Smith. 2009. Feature-rich translation by quasi-synchronous lattice parsing. In *EMNLP*.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *EMNLP*.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11:311–325.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *ACL*, pages 128–135.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*, pages 479–486.

Philipp Koehn. 2002. Europarl: A multilingual corpus for evaluation of machine translation. http://www.iccs.informatics.ed.ac.uk/~pkoehn/-publications/europarl.ps.

Jonas Kuhn. 2004. Experiments in parallel-text based grammar induction. In *ACL*, pages 470–477.

André F. T. Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *EMNLP*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *ACL*, pages 337–344.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *ACL*, pages 91–98.

Noah A. Smith and Jason Eisner. 2005. Guiding unsupervised grammar induction using contrastive estimation. In *International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Grammatical Inference Applications*, Edinburgh, July.

David A. Smith and Jason Eisner. 2006a. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*, pages 23–30.

Noah A. Smith and Jason Eisner. 2006b. Annealing structural bias in multilingual weighted grammar induction. In *ACL-COLING*, pages 569–576.

David A. Smith and Jason Eisner. 2007. Bootstrapping feature-rich dependency parsers with entropic priors. In *EMNLP-CoNLL*, pages 667–677.

David A. Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *EMNLP*, pages 145–156.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *EMNLP*, pages 49–56.

Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *ACL-IJCNLP*.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *EMNLP-CoNLL*, pages 22–32.

Self-Training PCFG Grammars with Latent Annotations Across Languages

Zhongqiang Huang[1]
[1]Laboratory for Computational Linguistics and Information Processing
Institute for Advanced Computer Studies
University of Maryland, College Park
zqhuang@umiacs.umd.edu

Mary Harper[1,2]
[2]Human Language Technology Center of Excellence
Johns Hopkins University
mharper@umiacs.umd.edu

Abstract

We investigate the effectiveness of self-training PCFG grammars with latent annotations (PCFG-LA) for parsing languages with different amounts of labeled training data. Compared to Charniak's lexicalized parser, the PCFG-LA parser was more effectively adapted to a language for which parsing has been less well developed (i.e., Chinese) and benefited more from self-training. We show for the first time that self-training is able to significantly improve the performance of the PCFG-LA parser, a single generative parser, on both small and large amounts of labeled training data. Our approach achieves state-of-the-art parsing accuracies for a single parser on both English (91.5%) and Chinese (85.2%).

1 Introduction

There is an extensive research literature on building high quality parsers for English (Collins, 1999; Charniak, 2000; Charniak and Johnson, 2005; Petrov et al., 2006), however, models for parsing other languages are less well developed. Take Chinese for example; there have been several attempts to develop accurate parsers for Chinese (Bikel and Chiang, 2000; Levy and Manning, 2003; Petrov and Klein, 2007), but the state-of-the-art performance, around 83% F measure on Penn Chinese Treebank (achieved by the Berkeley parser (Petrov and Klein, 2007)) falls far short of performance on English (∼90-92%). As pointed out in (Levy and Manning, 2003), there are many linguistic differences between Chinese and English, as well as structural differences between their corresponding treebanks, and some of these make it a harder task to parse Chinese. Additionally, the fact that the available treebanked Chinese materials are more limited than for English also increases the challenge of building high quality Chinese parsers. Many of these differences would also tend to apply to other less well investigated languages.

In this paper, we focus on English and Chinese because the former is a language for which extensive parsing research has been conducted while the latter is a language that has been less extensively studied. We adapt and improve the Berkeley parser, which learns PCFG grammars with latent annotations, and show through comparative studies that this parser significantly outperforms Charniak's parser, which was initially developed for English and subsequently ported to Chinese. We focus on answering two questions: how well does a parser perform across languages and how much does it benefit from self-training?

The first question is of special interest when choosing a parser that is designed for one language and adapting it to another less studied language. We improve the PCFG-LA parser by adding a language-independent method for handling rare words and adapt it to another language, Chinese, by creating a method to better model Chinese unknown words. Our results show that the PCFG-LA parser performs significantly better than Charniak's parser on Chinese, and is also somewhat more accurate on English, although both parsers have high accuracy.

The second question is important because labeled training data is often quite limited, especially for less well investigated languages, while unlabeled data is ubiquitous. Early investigations on self-training for parsing have had mixed results. Charniak (1997) reported no improvements from self-training a PCFG parser on the standard WSJ training set. Steedman et al. (2003) reported some degradation using a lexicalized tree adjoining grammar parser and minor improvement using Collins lexicalized PCFG parser; however, this gain was obtained only when the parser

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 832–841,
Singapore, 6-7 August 2009.

was trained on a small labeled set. Reichart and Rappoport (2007) obtained significant gains using Collins lexicalized parser with a different self-training protocol, but again they only looked at small labeled sets. McClosky et al. (2006) effectively utilized unlabeled data to improve parsing accuracy on the standard WSJ training set, but they used a two-stage parser comprised of Charniak's lexicalized probabilistic parser with n-best parsing and a discriminative reranking parser (Charniak and Johnson, 2005), and thus it would be better categorized as "co-training" (McClosky et al., 2008). It is worth noting that their attempts at self-training Charniak's lexicalized parser directly resulted in no improvement. There are other successful semi-supervised training approaches for dependency parsing, such as (Koo et al., 2008; Wang et al., 2008), and it would be interesting to investigate how they could be applied to constituency parsing.

We show in this paper, for the first time, that self-training is able to significantly improve the performance of the PCFG-LA parser, a single generative parser, on both small and large amounts of labeled training data, for both English and Chinese. With self-training, a fraction of the WSJ or CTB6 treebank training data is sufficient to train a PCFG-LA parser that is able to achieve or even beat the accuracies obtained using a single parser trained on the entire treebank without self-training. We conjecture based on our comparison of the PCFG-LA parser to Charniak's parser that the addition of self-training data helps the former parser learn more fine-grained latent annotations without over-fitting.

The rest of this paper is organized as follows. We describe the PCFG-LA parser and several enhancements in Section 2, and discuss self-training in Section 3. We then outline the experimental setup in Section 4, describe the results in Section 5, and present a detailed analysis in Section 6. The last section draws conclusions and describes future work.

2 Parsing Model

The Berkeley parser (Petrov et al., 2006; Petrov and Klein, 2007) is an efficient and effective parser that introduces latent annotations (Matsuzaki et al., 2005) to refine syntactic categories to learn better PCFG grammars. In the example parse tree in Figure 1(a), each syntactic category is split into multiple latent subcategories, and accordingly the original parse tree is decomposed into many parse trees with latent annotations. Figure 1(b) depicts one of such trees. The grammar and lexical rules are split accordingly, e.g., NP→PRP is split into different NP-i→PRP-j rules. The expansion probabilities of these split rules are the parameters of a PCFG-LA grammar.

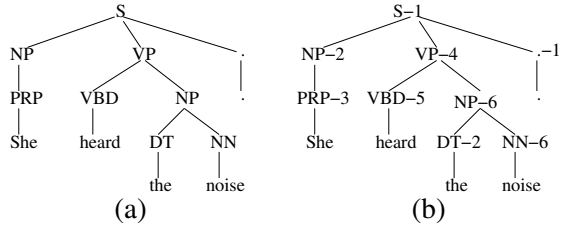


Figure 1: (a) original treebank tree, (b) after latent annotation.

The objective of training is to learn a grammar with latent annotations that maximizes the likelihood of the training trees, i.e., the sum of the likelihood of all parse trees with latent annotations. Since the latent annotations are not available in the treebank, a variant of the EM algorithm is utilized to learn the rule probabilities for them. The Berkeley parser employs a hierarchical split-merge method that gradually increases the number of latent annotations and adaptively allocates them to different treebank categories to best model the training data. In this paper, we call a grammar trained after n split-merge steps an n-th *order* grammar. The order of a grammar is a step (not continuous) function of the number of latent annotations because the split-merge algorithm first splits each latent annotation into two and then merges some of the splits back based on their ability to increase training likelihood.

For this paper, we implemented[1] our own version of Berkeley parser. Updates include better handling of rare words across languages, as well as unknown Chinese words. The parser is able to process difficult sentences robustly using adaptive beam expansion. The training algorithm was updated to support a wide range of self-training experiments (e.g., posterior-weighted unlabeled data, introducing self-training in later iterations) and to make use of multiple processors to parallelize EM training. The parallelization is crucial

[1] A major motivation for this implementation was to support some algorithms we are developing. Most of our enhancements will be merged with a future release of the Berkeley parser.

for training a model with large volumes of data in a reasonable amount of time[2].

We next describe the language-independent method to handle rare words, which is important for training better PCFG-LA grammars especially when the training data is limited in size, and our unknown Chinese word handling method, highlighting the importance of utilizing language-specific features to enhance parsing performance. As we will see later, both of these methods significantly improve parsing performance.

2.1 Rare Word Handling

Whereas rule expansions are frequently observed in the treebank, word-tag co-occurrences are sparser and more likely to suffer from over-fitting. Although the lexicon smoothing method in the Berkeley parser is able to make the word emission probabilities of different latent states of a POS tag more alike, the EM training algorithm still strongly discriminates among word identities. Suppose word tag pairs $\langle w_1, t\rangle$ and $\langle w_2, t\rangle$ both appear the same number of times in the training data. In a PCFG grammar without latent annotations, the probabilities of emitting these two words given tag t would be the same, i.e., $p(w_1|t) = p(w_2|t)$. After introducing latent annotation x to tag t, the emission probabilities of these two words given a latent state t_x may no longer be the same because $p(w_1|t_x)$ and $p(w_2|t_x)$ are two independent parameters that the EM algorithm optimizes on. It is beneficial to learn subcategories of POS tags to model different types of words, especially for frequent words; however, it is not desirable to strongly discriminate among rare words because it could distract the model from learning about common phenomena.

To handle this problem, the probability of a latent state t_x generating a rare word w is forced to be proportional to the emission probability of word w given the surface tag t. This is achieved by mapping all words with frequency less than threshold[3] λ to the *unk* symbol, and for each latent state t_x of a POS tag t, accumulating the word tag statistics of these rare words to $c_r(t_x, \textit{unk}) = \sum_{w:c(w)<\lambda} c(t_x, w)$, and then redistributing them among the rare words to estimate their emission probabilities:

$$c(t_x, w) = c_r(t_x, unk) \cdot \frac{c(t, w)}{c_r(t, unk)}$$
$$p(w|t_x) = c(t_x, w) / \sum\nolimits_w c(t_x, w)$$

2.2 Chinese Unknown Word Handling

The Berkeley parser utilizes statistics associated with rare words (e.g., suffix, capitalization) to estimate the emission probabilities of unknown words at decoding time. This is adequate for for English, however, only a limited number of classes of unknown words, such as digits and dates, are handled for Chinese. In this paper, we develop a character-based unknown word model inspired by (Huang et al., 2007) that reflects the fact that characters in any position (prefix, infix, or suffix) can be predictive of the part-of-speech (POS) type for Chinese words. In our model, the word emission probability, $p(w|t_x)$, of an unknown word w given the latent state t_x of POS tag t is estimated by the geometric average of the emission probability of the characters c_k in the word:

$$P(w|t_x) = \sqrt[n]{\prod\nolimits_{c_k \in w, P(c_k|t) \neq 0} P(c_k|t)}$$

where $n = |\{c_k \in w | P(c_k|t) \neq 0\}|$. Characters not seen in the training data are ignored in the computation of the geometric average. We back off to use the rare word statistics regardless of word identity when the above equation cannot be used to compute the emission probability.

3 Parser Self-Training

Our hypothesis is that combining automatically labeled parses with treebank trees will help the EM training of the PCFG-LA parser to make more informed decisions about latent annotations and thus generate more effective grammars. In this section, we discuss how self-training is applied to train a PCFG-LA parser.

There are several ways to automatically label the data. A fairly standard method is to parse the unlabeled sentences with a parser trained on labeled training data, and then combine the resulting parses with the treebank training data to retrain the parser. This is the approach we chose for self-training. An alternative approach is to run EM directly on the labeled treebank trees and the unlabeled sentences, without explicit parse trees for the unlabeled sentences. However, because the

[2]The parallel version is able to train our largest grammar on a 8-core machine within a week, while the non-parallel version is not able to finish even after 3 weeks.

[3]The value of λ is tuned on the development set.

brackets would need to be determined for the unlabeled sentences together with the latent annotations, this would increase the running time from linear in the number of expansion rules to cubic in the length of the sentence.

Another important decision is how to weight the gold standard and automatically labeled data when training a new parser model. Errors in the automatically labeled data could limit the accuracy of the self-trained model, especially when there is a much greater quantity of automatically labeled data than the gold standard training data. To balance the gold standard and automatically labeled data, one could duplicate the treebank data to match the size of the automatically labeled data; however, the training of the PCFG-LA parser would result in redundant applications of EM computations over the same data, increasing the cost of training. Instead we weight the posterior probabilities computed for the gold and automatically labeled data, so that they contribute equally to the resulting grammar. Our preliminary experiments show that balanced weighting is effective, especially for Chinese (about 0.4% absolute improvement) where the automatic parse trees have a relatively lower accuracy.

The training procedure of the PCFG-LA parser gradually introduces more latent annotations during each split-merge stage, and the self-labeled data can be introduced at any of these stages. Introduction of the self-labeled data in later stages, after some important annotations are learned from the treebank, could result in more effective learning. We have found that a middle stage introduction (after 3 split-merge iterations) of the automatically labeled data has an effect similar to balancing the weights of the gold and automatically labeled trees, possibly due to the fact that both methods place greater trust in the former than the latter. In this study, we introduce the automatically labeled data at the outset and weight it equally with the gold treebank training data in order to focus our experiments to support a deeper analysis.

4 Experimental Setup

For the English experiments, sections from the WSJ Penn Treebank are used as labeled training data: section 2-19 for training, section 22 for development, and section 23 as the test set. We also used 210k[4] sentences of unlabeled news articles in the BLLIP corpus for English self-training.

For the Chinese experiments, the Penn Chinese Treebank 6.0 (CTB6) (Xue et al., 2005) is used as labeled data. CTB6 includes both news articles and transcripts of broadcast news. We partitioned the news articles into train/development/test sets following Huang et al. (2007). The broadcast news section is added to the training data because it shares many of the characteristics of newswire text (e.g., fully punctuated, contains nonverbal expressions such as numbers and symbols). In addition, 210k sentences of unlabeled Chinese news articles are used for self-training. Since the Chinese parsers in our experiments require word-segmented sentences as input, the unlabeled sentences need to be word-segmented first. As shown in (Harper and Huang, 2009), the accuracy of automatic word segmentation has a great impact on Chinese parsing performance. We chose to use the Stanford segmenter (Chang et al., 2008) in our experiments because it is consistent with the treebank segmentation and provides the best performance among the segmenters that were tested. To minimize the discrepancy between the self-training data and the treebank data, we normalize both CTB6 and the self-training data using UW Decatur (Zhang and Kahn, 2008) text normalization.

Table 1 summarizes the data set sizes used in our experiments. We used slightly modified versions of the treebanks; empty nodes and nonterminal-yield unary rules[5], e.g., NP→VP, are deleted using tsurgeon (Levy and Andrew, 2006).

	Train	**Dev**	**Test**	**Unlabeled**
English	39.8k (950.0k)	1.7k (40.1k)	2.4k (56.7k)	210k (5,082.1k)
Chinese	24.4k (678.8k)	1.9k (51.2k)	2.0k (52.9k)	210k (6,254.9k)

Table 1: The number of sentences (and words in parentheses) in our experiments.

We trained parsers on 20%, 40%, 60%, 80%, and 100% of the treebank training data to evaluate

[4]This amount was constrained based on both CPU and memory. We plan to investigate cloud computing to exploit more unlabeled data.

[5]As nonterminal-yield unary rules are less likely to be posited by a statistical parser, it is common for parsers trained on the standard Chinese treebank to have substantially higher precision than recall. This gap between bracket recall and precision is alleviated without loss of parse accuracy by deleting the nonterminal-yield unary rules. This modification similarly benefits both parsers we study here.

the effect of the amount of labeled training data on parsing performance. We also compare how self-training impacts the models trained with different amounts of gold-standard training data. This allows us to simulate scenarios where the language has limited human-labeled resources.

We compare models trained only on the gold labeled training data with those that utilize additional unlabeled data. Self-training (PCFG-LA or Charniak) proceeds in two steps. In the first step, the parser is first trained on the allocated labeled training data (e.g., 40%) and is then used to parse the unlabeled data. In the second step, a new parser is trained on the weighted combination[6] of the allocated labeled training data and the additional automatically labeled data. The development set is used in each step to select the grammar order with the best accuracy for the PCFG-LA parser and to tune the smoothing parameters for Charniak's parser.

5 Results

In this section, we first present the effect of unknown and rare word handling for the PCFG-LA parser, and then compare and discuss the performance of the PCFG-LA parser and Charniak's parser across languages with different amounts of labeled training, either with or without self-training.

5.1 Rare and Unknown Word Handling

Table 2 reports the effect of unknown and rare word handing for the PCFG-LA parser trained on 100%[7] of the labeled training data. The rare word handling improves the English parser by 0.68% and the Chinese parser by 0.56% over the Berkeley parser. The Chinese unknown word handling method alone improves the Chinese parser by 0.47%. The rare and unknown handling methods together improve the Chinese parser by 0.92%. All the improvements are statistically significant[8].

We found that the rare word handling method becomes more effective as the number of latent annotations increases, especially when there is not a sufficient amount of labeled training data. Sharing statistics of the rare words during training results in more robust grammars with better parsing performance. The unknown word handling method also gives greater improvements on grammars trained on smaller amounts of training data, suggesting that it is quite helpful for modeling unseen words at decoding time. However, it tends to be less effective when the number of latent annotations increases, probably because the probability estimation of unseen words based on surface tags is less reliable for finer-gained latent annotations.

	English	Chinese
PCFG-LA	89.95	83.23
+R	90.63	83.79
+U	N/A	83.70
+R+U	N/A	84.15

Table 2: Effects of rare word handling (+R) and Chinese unknown handling (+U) on the test set.

5.2 Labeled Data Only

When comparing the two parsers on both languages in Figure 2 with treebank training, it is clear that they perform much better on English than Chinese. While this is probably due in part to the years of research on English, Chinese still appears to be more challenging than English. The comparison between the two parsing approaches provides two interesting conclusions.

First, the PCFG-LA parser always performs significantly better than Charniak's parser on Chinese, although both model English well. Admittedly Charniak's parser has not been optimized[9] on Chinese, but neither has the PCFG-LA parser[10]. The lexicalized model in Charniak's parser was first optimized for English and required sophisticated smoothing to deal with sparseness; however, the lexicalized model developed for Chinese works less well. In contrast, the PCFG-LA parser learns the latent annotations from the data, without any specification of what precisely should be modeled and how it should be modeled. This flexibility may help it better model new languages.

Second, while both parsers benefit from increased amounts of gold standard training data, the PCFG-LA parser gains more. The PCFG-LA parser is initially poorer than Charniak's parser

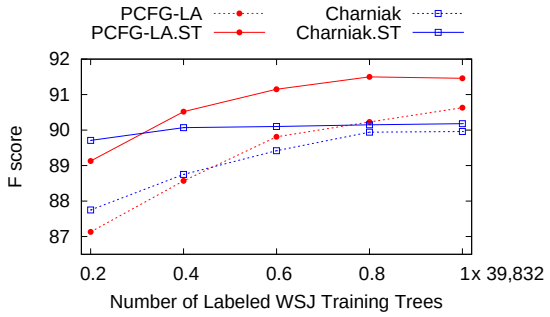
[6] We balance the size of manually and automatically labeled data by posterior weighting for the PCFG-LA parsers and by duplication for Charniak's parser.

[7] Greater improvements are obtained using smaller amounts of labeled training data.

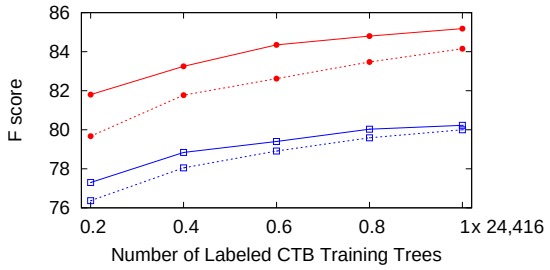
[8] We use Bikel's randomized parsing evaluation comparator to determine the significance ($p < 0.05$) of difference between two parsers' output.

[9] The Chinese port includes modification of the head table, implementation of a Chinese punctuation model, etc.

[10] The PCFG-LA parser without the unknown word handling method still outperforms Charniak's parser on Chinese.



(a) English



(b) Chinese

Figure 2: The performance of the PCFG-LA parser and Charniak's parser evaluated on the test set, trained with different amounts of labeled training data, with and without self-training (ST).

when trained on 20% WSJ training data, probably because the training data is too small for it to learn fine-grained annotations without over-fitting. As more labeled training data becomes available, the performance of the PCFG-LA parser improves quickly and finally outperforms Charniak's parser significantly. Moreover, performance of the PCFG-LA parser continues to grow when more labeled training data is available, while the performance of Charniak's parser levels out at around 80% of the labeled data. The PCFG-LA parser improves by 3.5% when moving from 20% to 100% training data, compared to a 2.21% gain for Charniak's parser. Similarly for Chinese, the PCFG-LA parser also gains more (4.48% vs 3.63%).

5.3 Labeled + Self-Labeled

The PCFG-LA parser is also able to benefit more from self-training than Charniak's parser. On the WSJ data set, Charniak's parser benefits from self-training initially when there is little labeled training data, but the improvement levels out quickly as more labeled training trees become available. In contrast, the PCFG-LA parser benefits consistently from self-training[11], even when using 100% of the labeled training set. Similar trends are also found for Chinese.

It should be noted that the PCFG-LA parser trained on a fraction of the treebank training data plus a large amount of self-labeled training data, which comes with little or no cost, performs comparably or even better than grammars trained with additional labeled training data. For example, the self-trained PCFG-LA parser with 60% labeled data is able to outperform the grammar trained with 100% labeled training data alone for both English and Chinese. With self-training, even 40% labeled WSJ training data is sufficient to train a PCFG-LA parser that is comparable to the model trained on the entire WSJ training data alone. This is of significant importance, especially for languages with limited human-labeled resources.

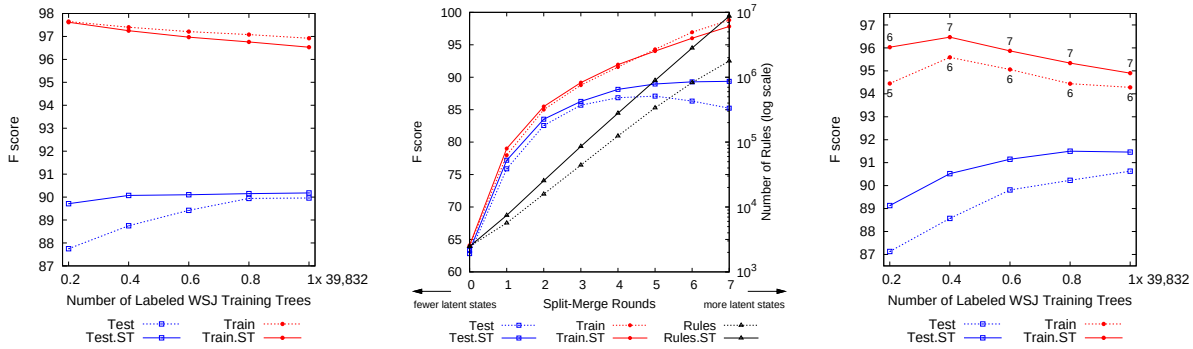
One might conjecture that the PCFG-LA parser benefits more from self-training than Charniak's parser because its self-labeled data has higher accuracy. However, this is not true. As shown in Figure 2 (a), the PCFG-LA parser trained with 40% of the WSJ training set alone has a much lower performance (88.57% vs 89.96%) than Charniak's parser trained on the full WSJ training set. With the same amount of self-training data (labeled by each parser), the resulting PCFG-LA parser obtains a much higher F score than the self-trained Charniak's parser (90.52% vs 90.18%). Similar patterns can also be found for Chinese.

	English	Chinese
PCFG-LA	90.63	84.15
+ Self-training	91.46	85.18

Table 3: Final results on the test set.

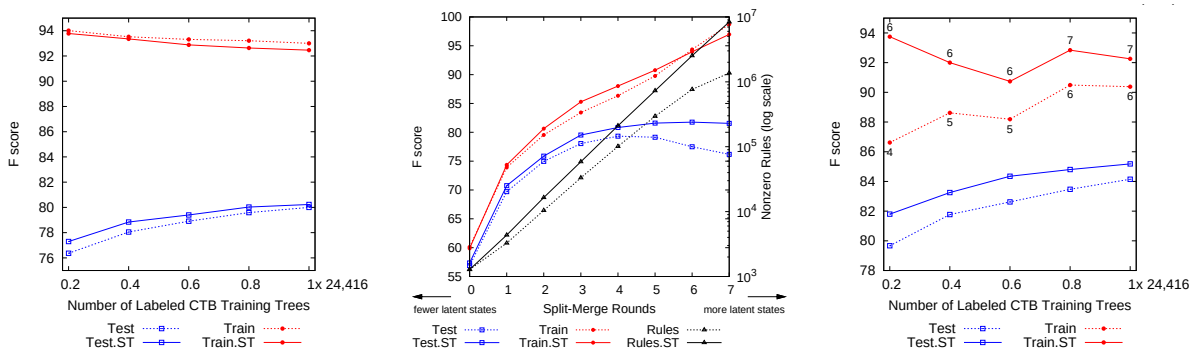
Table 3 reports the final results on the test set when trained on the entire WSJ or CTB6 training set. For English, self-training contributes 0.83% absolute improvement to the PCFG-LA parser, which is comparable to the improvement obtained from using semi-supervised training with the two-stage parser in (McClosky et al., 2006). Note that their improvement is achieved with the addition of 2,000k unlabeled sentences using the combination of a generative parser and a discriminative reranker, compared to using only 210k unlabeled sentences with a single generative parser in our approach. For Chinese, self-training results in a

[11] One may notice that the self-trained PCFG-LA parser with 100% labeled WSJ data has a slightly lower test accuracy than the self-trained PCFG-LA parser with 80% labeled WSJ data. This is due to the variance in parser performance when initialized with different seeds and the fact that the development set is used to pick the best model for evaluation.



(a) Charniak (b) PCFG-LA (20% WSJ) (c) PCFG-LA

Figure 3: (a) The training/test accuracy of Charniak's parser trained on varying amounts of labeled WSJ training data, with and without self-training (ST). (b) The training/test accuracy and the number of nonzero rules of the PCFG-LA grammars trained on 20% of the labeled WSJ training data, w/ and w/o ST. (c) The training/test accuracy of the PCFG-LA parser trained on varying amount of labeled WSJ training data, w/ and w/o ST; the numbers along the training curves indicate the order of the grammars.



(a) Charniak (b) PCFG-LA (20% CTB) (c) PCFG-LA

Figure 4: (a) The training/test accuracy of Charniak's parser trained on varying amounts of labeled CTB training data, with and without self-training (ST). (b) The training/test accuracy and the number of nonzero rules of the PCFG-LA grammars trained on 20% of the labeled CTB training data, w/ and w/o ST. (c) The training/test accuracy of the PCFG-LA parser trained on varying amount of labeled CTB training data, w/ and w/o ST; the numbers along the training curves indicate the order of the grammars.

state-of-the-art parsing model with 85.18% accuracy (1.03% absolute improvement) on a representative test set. Both improvements are statistically significant.

6 Analysis

In this section, we perform a series of analyses, focusing on English (refer to Figure 3), to investigate why the PCFG-LA parser benefits more from additional data, most particularly automatically labeled data, when compared to Charniak's parser. Similar analyses have been done for Chinese with similar results (refer to Figure 4).

Charniak's parser is a lexicalized PCFG parser that models lexicalized dependencies explicitly observable in the training data and relies on smoothing to avoid over-fitting. Although it is able to benefit from more training data because of broader lexicon and rule coverage and more robust estimation of parameters, its ability to benefit from the additional data is limited in the sense that it is not able to generate additional predictive features that are supported by this data. As shown in figure 3(a), the parsing accuracy of Charniak's parser on the test set improves as the amount of labeled training data increases; however, the training accuracy[12] degrades as more data is added. Note that the training accuracy[13] of Charniak's parser also

[12]The parser is tested on the treebank labeled set that the parser is trained on.

[13]The self-training data is combined with the labeled treebank trees in a weighted manner; otherwise, the training accuracy would be even lower.

decreases after the addition of self-training data. This is expected for models like Charniak's parser with fixed model parameters; it is harder to model more data with greater diversity. The addition of self-labeled data helps on the test set initially but it provides little gain when the labeled training data becomes relatively large.

In contrast, the PCFG-LA grammar is able to model the training data with different granularities. Fewer latent annotations are employed when the training set is small. As the size of the training data increases, it is able to allocate more latent annotations to better model the data. As shown in Figure 3 (b), for a fixed amount (20%) of labeled training data, the accuracy of the model on training data continues to improve as the number of latent annotation increases. Although it is important to limit the number of latent annotations to avoid over-fitting, the ability to model training data accurately given sufficient latent annotations is desirable when more training data is available. When trained on the labeled data (20%) alone, the 5-th order grammar achieves its optimal generalization performance (based on the development set) and begins to degrade afterwords. With the addition of self-training data, the 5-th order grammar achieves an even greater accuracy on the test set and its performance continues to increase[14] when moving to the 6-th or even 7-th order grammar.

Figure 3 (c) plots the training and test curves of the English PCFG-LA parser with varying amounts of labeled training data, with and without self-training. This figure differs substantially from Figure 3 (a). First, as mentioned earlier, the PCFG-LA parser benefits much more from self-training than Charniak's parser with moderate to large amounts of labeled training data. Second, in contrast to Charniak's parser for which training accuracy degrades consistently as the amount of labeled training data increases, the training accuracy of the PCFG-LA parser sometimes improves when trained on more labeled training data (e.g., the best model (at order 6) trained on 40%[15] labeled training data alone has a higher training accuracy than the best model (at order 5) trained on 20% labeled training data). Third, the addition of self-labeled data supports more accurate PCFG-LA grammars with higher orders than those trained without self-training, as evidenced by scores on both the training and test data. This suggests that the self-trained grammars are able to utilize more latent annotations to learn deeper dependencies.

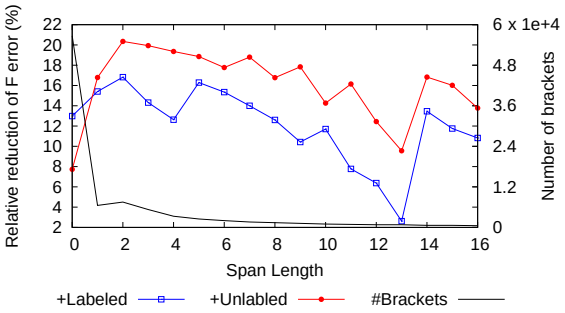


Figure 5: The relative reduction of bracketing errors for different span lengths, evaluated on the test set. The baseline model is the PCFG-LA parser trained on 20% of the WSJ training data. The +Unlabeled curve corresponds to the parser trained with the additional automatically labeled data and the +Labeled curve corresponds to the parser trained with additional 20% labeled training data. The counts of the brackets are computed on the gold reference. Span length '0' is designated for the effect on preterminal POS tags to differentiate it from the non-terminal brackets spanning only one word.

Figure 5 compares the effect of additional treebank labeled and automatically labeled data on the relative reduction of bracketing errors for different span lengths. It is clear from the figure that the improvement in parsing accuracy from self-training is the result of better bracketing across all span lengths[16]. However, even though the automatically labeled training data provides more improvement than the additional treebank labeled data in terms of parsing accuracy, this data is less effective at improving tagging accuracy than the additional treebank labeled training data.

So, how could self-training improve rule estimation when training the PCFG-LA parser with more latent annotations? One possibility is that the automatically labeled data smooths the parameter

[14] Although the 20% self-trained grammar has a higher test accuracy at the 7-th round than the 6-th round, the development accuracy was better at the 6-th round, and thus we report the test accuracy of the 6-th round grammar in Figure 3 (c).

[15] For models trained with greater amounts of labeled training data, although their training accuracy becomes lower (due to greater diversity) for the grammars (all at order 6) selected by the development set, their 7-th order grammars (not reported in the figure) actually have both higher training and test accuracies than the 6-th order grammar trained on less training data.

[16] There is a slight degradation in bracketing accuracy for some spans longer than 16 words, but the effect is negligible due to their low counts.

estimates in the EM algorithm, enabling effective training of models with more parameters to learn deeper dependencies. Let $p(a \rightarrow b|e,t)$ be the posterior probability of expanding subcategories a to b given the event e, which is a rule expansion on a treebank parse tree t. T_l and T_u are the sets of gold and automatically labeled parse trees, respectively. The update of the rule expansion probability $p(a \rightarrow b)$ in self-training (with weighting parameter α) can be expressed as:

$$\frac{\sum_{t \in T_l} \sum_{e \in t} p(a \rightarrow b|e,t) + \alpha \sum_{t \in T_u} \sum_{e \in t} p(a \rightarrow b|e,t)}{\sum_{b} (\sum_{t \in T_l} \sum_{e \in t} p(a \rightarrow b|e,t) + \alpha \sum_{t \in T_u} \sum_{e \in t} p(a \rightarrow b|e,t))}$$

Since the unlabeled data is parsed by a lower order grammar (with fewer latent annotations), the expected counts from the automatically labeled data can be thought of as counts from a lower-order grammar[17] that smooth the higher-order (with more latent annotations) grammar.

We observe that many of the rule parameters of the grammar trained on WSJ training data alone have zero probabilities (rules with extremely low probabilities are also filtered to zero), as was also pointed out in (Petrov et al., 2006). On the one hand, this is what we want because the grammar should learn to avoid impossible rule expansions. On the other hand, this might also be a sign of over-fitting of the labeled training data. As shown in Figure 3 (b), the grammar obtained with the addition of automatically labeled data contains many more non-zero rules, and its performance continues to improve with more latent annotations. Similar patterns also appear when using self-training for other amounts of labeled training data. As is partially reflected by the zero probability rules, the addition of the automatically labeled data enables the exploration of a broader parameter space with less danger of over-fitting the data. Also note that the benefit of the automatically labeled data is less clear in the early training stages (i.e., when there are fewer latent annotations), as can be seen in Figure 3 (b). This is probably because there is a small number of free parameters and the treebank data is sufficiently large for robust parameter estimation.

[17] We also trained models using only the automatically labeled data without combining it with human-labeled training data, but they were no more accurate than those trained on the human-labeled training data alone without self-training.

7 Conclusion

In this paper, we showed that PCFG-LA parsers can be more effectively applied to languages where parsing is less well developed and that they are able to benefit more from self-training than lexicalized generative parsers. We show for the first time that self-training is able to significantly improve the performance of a PCFG-LA parser, a single generative parser, on both small and large amounts of labeled training data.

We conjecture based on our analysis that the EM training algorithm is able to exploit the information available in both gold and automatically labeled data with more complex grammars while being less affected by over-fitting. Better results would be expected by combining the PCFG-LA parser with discriminative reranking approaches (Charniak and Johnson, 2005; Huang, 2008) for self training. Self-training should also benefit other discriminatively trained parsers with latent annotations (Petrov and Klein, 2008), although training would be much slower compared to using generative models, as in our case.

In future work, we plan to scale up the training process with more unlabeled training data (e.g., gigaword) and investigate automatic selection of materials that are most suitable for self-training. We also plan to investigate domain adaptation and apply the model to other languages with modest treebank resources. Finally, it is also important to explore other ways to exploit the use of unlabeled data.

Acknowledgments

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023 and NSF IIS-0703859. Any opinions, findings and/or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the institutions where the work was completed.

References

Daniel M. Bikel and David Chiang. 2000. Two statistical parsing models applied to the chinese treebank. In *Proceedings of the Second Chinese Language Processing Workshop*.

Pi-Chuan Chang, Michel Gally, and Christopher Manning. 2008. Optimizing chinese word segmentation

for machine translation performance. In *ACL 2008 Third Workshop on Statistical Machine Translation*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *ICAI*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *ACL*.

Michael John Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

Mary Harper and Zhongqiang Huang. 2009. Chinese statistical parsing. To appear in *The Gale Book*.

Zhongqiang Huang, Mary Harper, and Wen Wang. 2007. Mandarin part-of-speech tagging and discriminative reranking. In *EMNLP*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*.

Terry Koo, Xavier Carrera, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *ACL*.

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

Roger Levy and Christopher Manning. 2003. Is it harder to parse chinese, or the chinese treebank. In *ACL*.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *HLT-NAACL*.

David McClosky, Eugene Charniak, and Mark Johnson. 2008. When is self-training effective for parsing? In *COLING*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*.

Slav Petrov and Dan Klein. 2008. Sparse multi-scale grammars for discriminative latent variable parsing. In *EMNLP*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL*.

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

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *EACL*.

Qin Wang, Dale Schuurmans, and Dekang Lin. 2008. Semi-supervised convex training for dependency parsing. In *ACL*.

Nianwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*.

Bin Zhang and Jeremy G. Kahn. 2008. Evaluation of decatur text normalizer for language model training. Technical report, University of Washington.

An Alternative to Head-Driven Approaches for Parsing a (Relatively) Free Word-Order Language

Reut Tsarfaty **Khalil Sima'an** **Remko Scha**
Institute for Logic Language and Computation
University of Amsterdam
{r.tsarfaty,k.simaan,r.scha}@uva.nl

Abstract

Applying statistical parsers developed for English to languages with freer word-order has turned out to be harder than expected. This paper investigates the adequacy of different statistical parsing models for dealing with a (relatively) free word-order language. We show that the recently proposed *Relational-Realizational (RR)* model consistently outperforms state-of-the-art *Head-Driven (HD)* models on the Hebrew Treebank. Our analysis reveals a weakness of HD models: their intrinsic focus on configurational information. We conclude that the form-function separation ingrained in RR models makes them better suited for parsing nonconfigurational phenomena.

1 Introduction

Parsing technology has come a long way since Charniak (1996) demonstrated that a simple treebank PCFG performs better than any other parser (with $F_1 75$ accuracy) on parsing the WSJ Penn treebank (Marcus et al., 1993). Treebank Grammars (Scha, 1990; Charniak, 1996) trained on large corpora nowadays present the best available means to parse natural language text.

The performance curve for parsing the WSJ was a steep one at first, as the incorporation of notions such as *head*, *distance*, *subcategorization* (Charniak, 1997; Collins, 1999) brought about a dramatic increase in parsing accuracy to the level of $F_1 88$. Discriminative approaches, Data-Oriented Parsing ('all-subtrees') approaches, and self-training techniques brought further improvements, and recent results are starting to level off at around $F_1 92.1$ (McClosky et al., 2008).

As the interest of the NLP community grows to encompass more languages, we observe efforts towards adapting an English parser for parsing other languages (e.g., (Collins et al., 1999)), or towards designing a language-independent framework based on principles underlying the models for parsing English (Bikel, 2002). The performance curve for parsing other languages with these models looks rather different. A case in point is Modern Standard Arabic. Since the initial effort of (Bikel, 2002) to parse the Arabic treebank (Maamouri et al., 2004), which yielded $F_1 75$ accuracy, four years and successive revisions have led to no more than $F_1 79$ (Maamouri et al., 2008).

This pattern from Arabic is not peculiar. The level of state-of-the-art results for other languages still lags behind those for English, even after putting considerable effort into the adaptation.[1] Given that these languages are inherently different from English and from one another, it appears that we cannot avoid a question concerning the *adequacy* of the models used to parse them. That is, given the properties of a language, which modeling strategy would be appropriate for parsing it?

Until recently, there has been practically no computationally affordable alternative to the *Head-Driven (HD)* approach in the development of phrase-structure based statistical parsing models. Recently, we proposed the *Relational-Realizational (RR)* approach that rests upon different premises (Tsarfaty and Sima'an, 2008). The question of how the RR model fares against the HD models that have so far been predominantly used has never been tackled. Yet, it is precisely such a comparison that can shed new light on the question of adequacy we posed above.

Empirically quantifying the effects of different modeling choices has been addressed for English by, e.g., (Johnson, 1998; Klein and Manning, 2003), and for German by, e.g., (Dubey, 2004;

[1] Consider, e.g., "The PaGe shared task on parsing German" (Kubler, 2008), reporting $F_1 75, F_1 79, F_1 83$ for the participating parsers.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 842–851,
Singapore, 6-7 August 2009.

Rafferty and Manning, 2008). This paper provides an empirical systematic comparison of conceptually different modeling strategies with respect to parsing Hebrew. This comparison is intended to provide a first answer to the question of parser adequacy in the face of word-order freedom.

Our two empirical results are unequivocal. Firstly, RR models significantly outperform HD models (about 2 points absolute improvement in F_1) in parsing the Modern Hebrew treebank. In particular, RR models show better performance in identifying the constituents for which syntactic positions are relatively free. Secondly, we show a novel variation of the HD model, incorporating the *Relational* notions of the RR model, on the hypothesis that this might bridge the gap. The RR model remains superior.

Our post-experimental analysis shows that HD modeling is inherently problematic for parsing a language with freer word-order because of the hard-wiring of notions such as *left*, *right* and *distance from the head*. RR models, taking a principled approach towards capturing variable form-function correspondence patterns, are better suited for parsing *nonconfigurational* phenomena.

2 The Data

This section describes some properties of Modern Hebrew (henceforth, Hebrew) that make it significantly different from English. These properties affect the syntactic representations found in the Hebrew Treebank and the kind of syntactic phenomena a parser for Hebrew has to cope with.

Modern Hebrew is a Semitic language with a canonical SVO word-order pattern,[2] yet it allows considerable freedom in the placement of syntactic constituents in a clause. For example, linguistic elements of any kind may be fronted, triggering an inversion familiar from Germanic languages as in (1b) (*Triggered Inversion (TI)* in (Shlonsky, 1997)). Under some information structuring conditions, *Verb Initial (VI)* constructions are also allowed, as in (1c) (Melnik, 2002). All sentences in (1) thus mean "Dani gave the present to Dina", despite their different word-ordering.

(1) a. *dani natan et hamatana ledina*
Dani gave ACC the-present to-Dina

b. *et hamatana natan dani ledina*
ACC the-present gave Dani to-Dina

[2]SVO is an abbreviation for the Subject-Verb-Object type in the *basic word-order* typology of (Greenberg, 1963).

Word Order	Frequency	Relative Frequency
SV	1612	41%
VS	1144	29%
No S	624	16%
No V	550	14%

Table 1: **Modern Hebrew Predicative Clause-Types** in 3930 Predicative Matrix Clauses in the Training Set of the Modern Hebrew Treebank.

c. *natan dani et hamatana ledina*
gave Dani ACC the-present to-Dina

A corpus study we conducted on a fragment of the Modern Hebrew treebank reveals that although there is a significant number of subjects preceding verbs in simple (matrix) clauses (41%), there are also a fair number of sentences for which this order is reversed (29%), and there is evidence for other configurations, such as empty realization of subjects (16%) and non-verbal realization of predicates (14%).

In the face of such lack of consistency in its configurational position, the grammatical function *Object* in Hebrew is indicated by *Differential Object Marking (DOM)* (Aissen, 2003). NP objects in Hebrew are marked for *accusativity* (using the marker *et*) if they are also marked for *definiteness* (indicated by the prefix *ha*). So, in contrast with (2a)-(2b), the indefinite object renders (2c) ungrammatical, and the missing accusativity renders (2d) awkward. The fact that marking NP objects involves the joint contribution of multiple surface elements (*et, ha*) contributing features to the NP constituent is referred to as *extended exponence* (Matthews, 1993, p. 182).

(2) a. *dani natan matana ledina*
Dani gave present to-Dina
"Dani gave a present to Dina"

b. *dani natan et hamatana ledina*
Dani gave ACC the-present to-Dina
"Dani gave the present to Dina"

c. **dani natan et matana ledina*
Dani gave ACC present to-Dina

d. *??dani natan hamatana ledina*
Dani gave the-present to-Dina

These data pose a challenge to generative parsing models, as they would be required to generate alternative word-order patterns while maintaining a coherent pattern of object marking, encom-

passing the contribution of multiple surface exponents. The question this paper addresses is therefore what kind of modeling approach would be adequate for modeling the interplay between *syntax* and *morphology* in marking grammatical relations in Hebrew, as reflected by the sentence-pair (3). They both mean, roughly, "Dani gave the present to Dina yesterday; their word-order vary, but the pattern of object marking is retained.

(3) a. *dani natan etmol et hamatana ledina*
Dani gave yesterday ACC the-present to-Dina

b. *et hamatana natan etmol dani ledina*
ACC the-present gave yesterday dani to-dina

3 The Models

The different models we experiment with are all trained on syntactic structures annotated in the Modern Hebrew Treebank (Sima'an et al., 2001). The native representation of clause-level categories in the Treebank employs flat structures. This choice was made due to the lack of empirical evidence in Hebrew for grouping freely positioned syntactic elements to form a constituent.[3] In order to compensate for the ambiguity in the *interpretation* of flat structures, additional information such as morphological marking and grammatical function labels is added to the phrase-structure trees.

3.1 The *State-Splits* Approach

The simplest way to encode grammatical functions information on top of the phrase-structure representation in the treebank is by decorating non-terminal nodes with morphological or functional features, similarly to the rich representation format of syntactic categories in GPSG. This is the approach taken by the annotators of the Hebrew treebank in which information about morphological marking appears at multiple levels of constituency (Guthmann et al., 2009), and functional features (such as *subject*, *object*, etc.) decorate phrase-level constituent labels (Sima'an et al., 2001). The S-level representation of our example sentences (3a)–(3b) then would be as we depict in figure 1, which can be read off as feature-rich PCFG productions. We refer to this approach as the *State-Splits (SP)* approach, which serves as the baseline for the rest of our investigation.

3.2 The *Head-Driven* Approach

Following the linguistic wisdom that the internal organization of syntactic constituents revolves around their *heads*, *Head-Driven (HD)* models have been proposed by (Magerman, 1995; Charniak, 1997; Collins, 1999). In a generative HD model, the head daughter is generated first, conditioned on properties of the mother node. Then, sisters of the head daughter are generated conditioned on the head, typically by *left* and *right* generation processes. Overall, HD processes have the modeling advantage that they capture structurally-marked positions that characterize the *argument structure* of the sentence. The simplest possible process uses unigram probabilities, but (Klein and Manning, 2003) show that using *vertical* and *horizontal* Markovization improves parsing accuracy.[4]

An unlexicalized generative HD model will generate our two example sentences as we illustrate in figure 2. The generation of the context-free events in figure 1 is then broken down to seven different context-free parameters each, encoding head-parent and head-sister structural relationships — the latter mediated with a structurally-marked *delta* function (Δ_i). The rich morphological representation of phrase-level NP objects (+*def/acc*), for instance, is conditioned on the *head* sister, its *direction*, and the *distance from the head* (check, e.g., nodes $\Delta_{L_1}, \Delta_{R_2}$).

3.3 The *Relational-Realizational* Approach

The *Relational-Realizational (RR)* parsing model of (Tsarfaty and Sima'an, 2008) similarly decomposes the generation of the context-free events in figure 1 into multiple independent parameters, but does so in a conceptually different way. Instead of decomposing a context-free event to *head* and *sisters*, the RR model is best viewed as a generative grammar that decomposes it to *form* and *function*.

The RR grammar first generates a set of grammatical functions depicting the *Relational Network (RN)* (Perlmutter, 1982) of the clause. This

[3] Such clauses are defined formally as *exocentric* in formal theories of syntax, and are used to describe syntactic structures in, e.g., Tagalog, Hungarian and Warlpiri (Bresnan, 2001, page 110). This flat representation format is characteristic of treebanks for other languages with relatively-free word-order as well, such as German (cf. (Kubler, 2008)).

[4] The success of Head-Driven models (Charniak, 1997; Collins, 2003) was initially attributed to the fact that they were fully lexicalized, but (Klein and Manning, 2003) show that an unlexicalized model combining Head-Driven Markovian processes with linguistically motivated state-splits can approach the performance of fully lexicalized models.

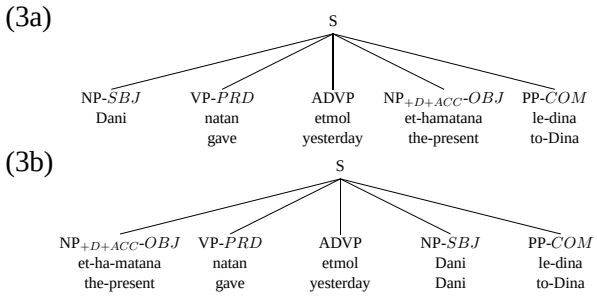


Figure 1: The *State-Splits* Approach for Ex. (3)

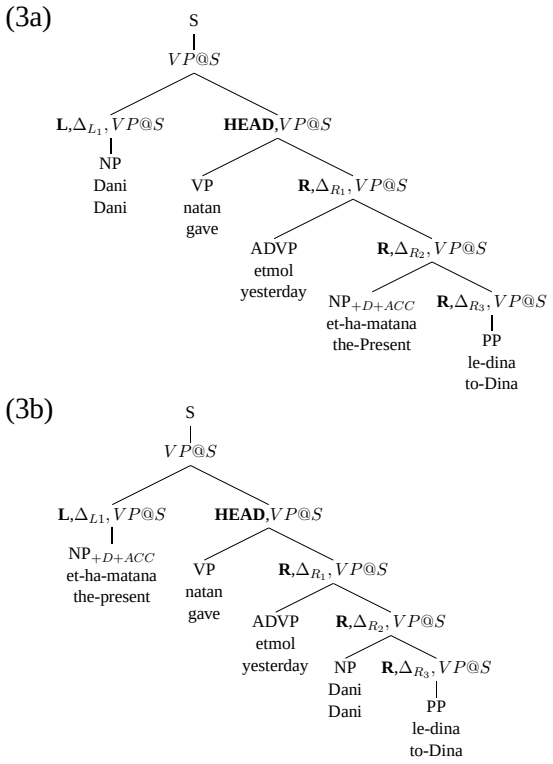


Figure 2: The *Head-Driven* Approach for Ex. (3)

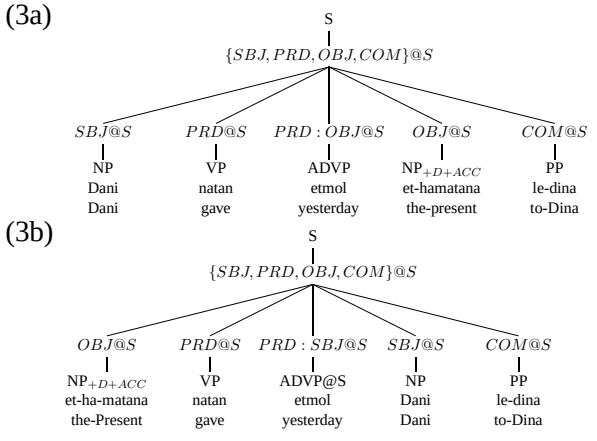


Figure 3: The *Relational-Realizational* Approach

RN provides an abstract set-theoretic representation of the *argument structure* of the clause.[5] This is called the *projection* phase. Then an ordering of the grammatical relations is generated, including reserved contextual slots for adjunction and/or punctuation marks. This is called the *configuration* phase. Finally, each of the grammatical function labels and adjunction slots gets realized as a morphosyntactic representation (a category label plus dominated morphological features) of the respective daughter constituent. This is called the *realization* phase.[6]

Figure 3 shows the generation of sentences (3a)–(3b) following the *projection*, *configuration* and *realization* phases corresponding to the top-down context-free layers of the tree. In both cases, the same relational network is generated, capturing the fact that they have the same argument structure. Then the different orderings of the grammatical elements are generated, reserving an adjunction slot for sentential modification (labeled by short context). Interestingly, the HD/RR models for our sentences are of comparable size (seven parameters) but the parameter types encode radically different notions. Illustrative of the difference is the realization of a morphologically marked NP object. In the RR model this is conditioned on a grammatical relation (check, for instance, node OBJ@S) and in the HD model it is conditioned on linear ordering or configurational notions such as *left*, *right* and *distance*.

4 Experiments

Goal We set out to compare the performance of the different modeling approaches for parsing Modern Hebrew. Considerable effort was devoted to making the models strictly comparable, in terms of preparing the data, defining statistical events, and unifying the rules determining cross-cutting linguistic notions (e.g., *heads* and *predicates*, *grammatical functions* and *subcat sets*). We spell out some of the setup considerations below.

Data We use the Modern Hebrew treebank (MHTB) (Sima'an et al., 2001) consisting of 6501 sentences from news-wire texts, morphologically analyzed and syntactically annotated as phrase-

[5] Unlike in HD models or dependency grammars, the *head* predicative element has no distinguished status here.

[6] Realization of adjunction slots (but not of function labels) may generate multiple sisters adjoining at a single position.

GF	*Description*	*Applicable to...*
PRD	Predicative Elements	VP, PREDP
SBJ	Grammatical Subjects	NP, SBAR
OBJ	Direct Objects	NP
COM	Indirect Objects	NP, PP
	FInite Complements	SBAR
IC	Infinitival Complements	VP
CNJ	A Conjunct within	
	a Conjunction Structure	All

Table 2: Grammatical Functions in the MHTB

SP-PCFG	Expansion	P($C_{l_n}, \ldots, C_h, \ldots, C_{r_n}|P$)
HD-PCFG	Head	P($C_h|P$)
	Left Branch?	P($\mathbf{L}$:Δ_{l_1}, $\mathbf{H}$:$\Delta_h|C_h, P$)
	Right Branch?	P(C_h, $\mathbf{R}$:$\Delta_{r_1}|\Delta_h, C_h, P$)
	Left Arg/Mod	P($C_{l_i}, \Delta_{l_{i+1}}|\ \mathbf{L}\ , \Delta_{l_i}, C_h, P$)
	Right Arg/Mod	P($C_{r_i}, \Delta_{r_{i+1}}|\ \mathbf{R}\ , \Delta_{r_i}, C_h, P$)
	Left Final?	P($C_1|\ \mathbf{L}\ , \Delta_{l_{n-1}}, C_h, P$)
	Right Final?	P($C_n|\ \mathbf{R}\ , \Delta_{r_{n-1}}, C_h, P$)
RR-PCFG	Projection	P($\{gr_1, \ldots, gr_m\}|P$)
	Configuration	P($\langle gr_1, \ldots, gr_m\rangle|\{gr_1, \ldots, gr_m\}P$)
	Realization	P($C_j|gr_j, P$)
	Adjunction	P($C_{j_1}, \ldots, C_{j_n}|gr_j : gr_{j+1}, P$)

Table 3: PCFG Parameter Classes for All Models

structure trees. In our version of the MHTB, *definiteness* and *accusativity* features are percolated from the PoS-tags level to phrase-level categories, extending the procedure of (Guthmann et al., 2009). For all models, we applied non-terminal state-splits distinguishing finite from non-finite verb forms and possessive from non-possessive noun phrases. We head-annotated the treebank, and based on the 'subject', 'object', 'complement' and 'conjunction' labels in the MHTB we devised an automatic procedure to annotate all the grammatical functions indicated in table 2.[7]

Procedure For all models, we learn a PCFG by reading off the parameters described in table 3, in accordance with the trees depicted in figures 1–3.[8] For all models, we use relative frequency estimates. For lexical parameters, we use a simple smoothing procedure assigning probability to unknown words using the per-tag distribution of rare words ("rare" threshold set to < 2). The input to our parser consists of morphologically segmented surface forms, and the parser has to assign the syntactic as well as morphological analysis to the surface segments.[9] We use the standard development/training/test split as in (Tsarfaty and Sima'an, 2008). Since our goal is a detailed comparison and fine-grained analysis of the results we concentrate on the development set. We use a general-purpose CKY parser (Schmid, 2004) to exhaustively parse the sentences, and we strip off all model-specific information prior to evaluation.

Evaluation We use standard *Parseval* measures calculated for the original, flat, canonical representation of the parse trees.[10] We report *Precision/Recall* for the coarse-grained non-terminal categories. In addition to overall Parseval scores we report the accuracy results *Per Syntactic Category*. We further report model size in terms of the number of parameters. As is well known in Machine Learning, models with more parameters require more data to learn, and are more vulnerable to sparseness. In our evaluation we thus follow the rule of thumb that (all else being equal) for models of equal size the better performing model is preferred, and for models with equal performance, the smaller one is preferred.

5 Results and Analysis

5.1 Overall Results

Table 4 shows the parsing results for the **State-Split (SP) PCFG**, the **Head-Driven (HD) PCFG** and the **Relational-Realizational (RR) PCFG** models on parsing the Modern Hebrew Treebank, with *definiteness* and *accusativity* marked on PoS-tags as well as phrase-level categories. For all models, we experiment with grandparent encoding. For non-HD models, we also examine the utility of a head-category split.[11]

[7] The enhanced corpus will be available at `www.science.uva.nl/~rtsarfat/resources.htm`.

[8] Our training procedure is strictly equivalent to the transform-detransform methodology of (Johnson, 1998), but we implement a tree-traverse procedure as in (Bikel, 2002) collecting all parameters per event at once.

[9] This setup is more difficult than, e.g., the Arabic parsing setup of (Bikel, 2002), as they assume gold-standard pos-tags as input. Yet it is easier than the setup of (Tsarfaty, 2006; Goldberg and Tsarfaty, 2008) which uses unsegmented surface forms as input. The decision to use segmented and untagged forms was made to retain a realistic scenario. Morphological analysis is known to be ambiguous, and we do not assume that morphological features are known up front. Morphological segmentation is also ambiguous, but for our purposes it is unavoidable. When comparing different models on an individual sentence they may propose segmentation to sequences of different lengths, for which accuracy results cannot be faithfully compared. See (Tsarfaty, 2006) for discussion.

[10] The flat canonical representation also allows for a fair comparison that is not biased by the differing branching factors of the different models.

[11] In HD models, a head-tag is already assumed in the conditioning context for sister nodes (Klein and Manning, 2003).

SP-PCFG				
Grand-Parent	−	−	+	+
Head-Tag	−	+	−	+
Prec/Rec	70.05/72.40	71.14/72.03	**74.66/74.35**	71.99/72.17
(#Params)	(4995)	(8366)	**(7385)**	(11633)
HD-PCFG				
Grand-Parent	−	−	+	+
Markov	0	1	0	1
Prec/Rec	66.87/71.64	70.40/74.35	73.04/71.94	**73.52/74.84**
(#Params)	(6678)	(10015)	(19066)	**(21399)**
RR-PCFG				
Grand-Parent	−	−	+	+
Head Tag	−	+	−	+
Prec/Rec	69.90/73.96	72.96/75.73	74.19/75.03	**76.32/76.51**
(#Params)	(3791)	(7546)	(7611)	**(13618)**

Table 4: **The Performance of Different Models in Parsing Hebrew:** Parsing Results Prec/Recall for Sentences of Length ≤ 40.

Model / Category	**SP-PCFG**	**HD-PCFG**	**RR-PCFG**
NP	77.39 / 74.32	77.94 / 73.75	**78.96 / 76.11**
PP	71.78 / 71.14	71.83 / 69.24	**74.4 / 72.02**
SBAR	55.73 / 59.71	53.79 / 57.49	**57.97 / 61.67**
ADVP	71.37 / 77.01	72.52 / 73.56	**73.57 / 77.59**
ADJP	**79.37 / 78.96**	78.47 / 77.14	78.69 / 78.18
S	**73.25 / 79.07**	71.07 / 76.49	72.37 / 78.33
SQ	36.00 / **32.14**	30.77 / 14.29	**55.56** / 17.86
PREDP	36.31 / 39.63	**44.74** / 39.63	44.51 / **46.95**
VP	76.34 / 80.81	77.33 / **82.51**	**78.59** / 81.18

Table 5: **Per-Category Evaluation of Parsing Performance for Different Models:** Prec/Rec Per Category Calculated for All Sentences.

For all models, grandparent encoding is helpful. For HD models, a higher Markovian order improves performance. This shows that even in Hebrew there are linear-precedence tendencies that help steer the disambiguation in the right direction, which is in line with our observation that word-order patterns in Modern Hebrew are not completely free (cf. table 1).

The best SP model performs equally or better than all HD models. This might be due to the smaller size of SP grammars, resulting in more robust estimates. But it is remarkable that, given the feature-rich representation, such a simple treebank grammar provides better disambiguation capacity than linguistically articulated HD models. We attribute this to the fact that parent-daughter relations have a stronger association with grammatical functions than relations between neighbouring nodes. For Hebrew, such adjacency relations may be arbitrary due to word-order variability.

Overall, RR models show the best performance for the set of all models with parent encoding, and for the set of all models without. Our best RR model shows 6.6%/8.4% Prec/Rec error reduction from the best SP model. The Recall improvement shows that the RR model is much better in generalizing, recovering successfully more of the constituents found in the gold representation. The best RR model also outperforms HD models with 8.7%/6.7% Prec/Rec error reduction from the best HD model. The resulting precision improvement of the RR relative to HD is larger than the improvement relative to SP, and the Recall improvement pattern is reversed. So it seems that the HD model generalizes better than the SP model, but also gets generalizations wrong more often than the SP model.

The RR model combines the generalization advantage of breaking down context-free events while it maintains the coherence advantage of learning flat trees (cf. (Johnson, 1998)). The best RR model obtains the best performance among all models: $F_1 76.41$. To put this result in context, for the setting in which the Arabic parser of (Maamouri et al., 2008) obtains $F_1 78.1$, — i.e., with gold standard feature-rich tags — the best RR model obtains $F_1 83.3$ accuracy which is the best parsing result reported for a Semitic language so far. RR models also have the advantage of resulting in more compact grammars, which makes learning and parsing with them much more computationally efficient.

5.2 Per-Category Break-Down Analysis

To understand better the merits of the different models we conducted a break-down analysis of performance-per-category for the best performing models of each kind. The break-down results are shown in table 5. We divided the table into three sets of categories: those for which the RR model gave the best performance, those for which the SP model gave the best performance, and those for which there is no clear trend.

The most striking outcome is that the RR model identifies at higher accuracy precisely those syntactic elements that are freely positioned with re-

In our SP or RR models, head-information is used as yet another feature-value pair rather than an object with a distinguished status during generation.

spect to the head: NPs, PPs, ADVPs and SBARs. Adjectives, in contrast, have clear ordering constraints — they always appear after the noun. S level elements, when embedded, always appear immediately after a conjunction or a relativizer. In particular, NPs and PPs realize arguments and adjuncts that may occupy different positions relative to the head. The RR model is better than the other models in identifying those elements partly because morphological information helps to disambiguate syntactically relevant chunks and make correct attachment decisions about them.

Remarkably, predicative (verb-less) phrases (PREDP), which are characteristic of Semitic languages, are hard to parse, but here too the RR does slightly better than the other two, as it allows for variability in the means to realize a (verbal or verb-less) predicate. Both RR and HD models outperform SP for VPs, which is due to the specific nature of VPs in the MHTB – they exist *only* for complement phrases with strict linear ordering.

6 Distances, Functions and Subcategorization Frames

Markovian processes to the *left* and to the *right* of the head provide a first approximation of the predicate's *argument structure*, as they capture trends in the co-occurrences of constituents reflected in their pattern of *positioning* and *adjacency*. But as our results so far show, such an approximation is empirically less rewarding for a language in which grammatical relations are not tightly correlated with structural notions.[12]

Collins (2003) attempted a more abstract formulation of argument-structure by articulating left and right *subcat-sets*. Each set represents those arguments that are expected to occur at each side of the head. Argument sisters ("complements") are generated if and only if they are required, and their generation 'cancels' the requirement in the set. Adjuncts ("modifiers") may be freely generated at any position.

At first glance, such a dissociation of configurational positions and subcategorization sets seems to be more adequate for parsing Hebrew, because it allows for some variability in the order of generation. But here too, since the model uses sets of

[12]Conditioning based on *adjacency* and *distance* is also common inside *dependency parsing* models, and we conjecture that this is one of the reasons for their difficulty in coping with freer word-order languages, a difficulty pointed out in (Nivre et al., 2007).

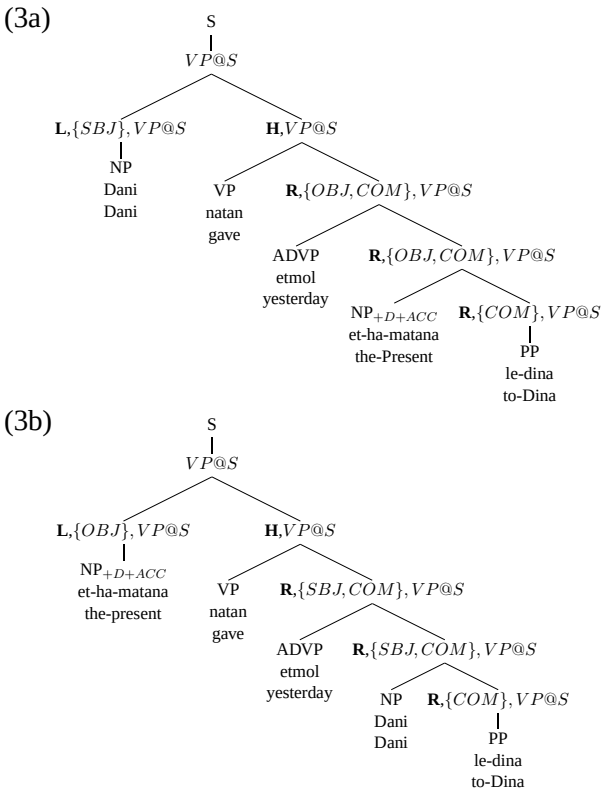


Figure 4: The *Relational Head-Driven* Approach

constituent labels, it disambiguates the grammatical functions of an NP solely based on the direction of the head, which is adequate for English but not for Hebrew. In order to relax this association further, we propose to replace constituent labels in the subcat-sets with grammatical relations identical to the functional elements in the relational network of the RR. This provides means to mediate the cancellation of constituents in the sets with their functions and correlate it with morphology.

To get an idea of the implications of such a modeling strategy, let us consider our example sentences in such a Relational-HD model as depicted in figure 4. Both representations share the event of generating the verbal head. Sisters are generated conditioned on the head and the functional elements remaining to be "cancelled". Each of the two trees consists of an event realizing an "object", one for an NP to the right of the head, and the other for an NP to its left. In both cases, an object constituent will be generated jointly with the morphological features associated with it. Evidently, when using sets of grammatical relations instead of constituent-labels, correlation of morphology and grammatical functions is more straight-forward to maintain.

Model	**SP-PCFG**	**HD-PCFG**	**HD-PCFG**	**HD-PCFG**	**HD-PCFG**	**RR-PCFG**
Type of Distance Δ *or Subcategorization*	Phrase-Level State-Splits	Intervening Verb/Punc	Left and Right #Constituents	Left and Right Constituent Labels	Left and Right Function Labels	Subcat Sets Configuration
Precision/Recall (#Params)	70.95/70.32 (13884)	72.39 / 71.97 (11650)	72.70 / 74.46 (18058)	72.42 / 74.29 (16334)	72.84/74.62 (16460)	76.32/76.51 (13618)

Table 6: **Incorporating Distance and Grammatical Functions into Head-Driven Parsing Models** Reporting Precison/Recall (#Parameters) for Sentences Length < 40.

6.1 Results and Analysis

Table 6 reports the results of experimenting with HD models with different instantiations of a *distance* function, starting from the standard notion of (Collins, 2003) and ending with our proposed, relational, function sets. For all HD models, we retain the *head*, *left* and *right* generation cycle and only change the conditioning context (Δ_i) for sister generation.

As a baseline, we show the results of adding grammatical function information as state-splits on top of an SP-PCFG.[13] This SP model presents much lower performance than the RR model although they are almost of the same size and they start off with the same information. This result shows that sophisticated modeling can blunt the claws of the sparseness problem. One may obtain the same number of parameters for two different models, but correlate them with more profound linguistic notions in one model than in the other. In our case, there is more statistical evidence in the data for, e.g., case marking patterns, than for association of grammatical relations with structurally-marked positions.

For all HD variations, the RR model continues to outperform HD models. The function-set variation performs slightly (but not significantly) better than the category-set. What seems to be still standing in the way of getting useful disambiguation cues for HD models is the fact that the *left* and *right* direction of realization is hardwired in their representation. This breaks down a coherent distribution over morphosyntactic representations realizing grammatical relations to arbitrary position-dependent fragments, which results in larger grammars and inferior performance.[14]

[13]The startegy of adding grammatical functions as state-splits is used in, e.g., German (Rafferty and Manning, 2008).

[14]Due to the difference in the size of the grammars, one could argue that smoothing will bridge the gap between the HD and RR modeling strategies. However, the better size/accuracy trade-off shown here for RR models suggests that they provide a good bias/variance balancing point, especially for feature-rich models characterizing morphologically rich languages. A promising strategy then would be to smooth or split-and-merge (Petrov et al., 2006)) RR-based models rather than to add an elaborate smoothing component to configurationally-based HD models.

7 A Typological Detour

Hebrew, Arabic and other Semitic Languages are known to be substantially different from English in that English is strongly *configurational*. In configurational languages word-order is fixed, and information about the grammatical functions of constituents (e.g., *subject* or *object*) is often correlated with structurally-marked positions inside highly-nested constituency structures. *Nonconfigurational* languages (Hale, 1983), in contrast, allow for freedom in their word-ordering and information about grammatical relations between constituents is often marked by means of *morphology*.

Configurationality is hardly a clear-cut notion. The difference in the configurationality level of different languages is often conceived as depicted in figure 7. In *linguistic typology*, the branch of linguistics that studies the differences between languages (Song, 2001), the division of labor between linear ordering and morphological marking in the realization of grammatical relations is often viewed as a continuum. Common wisdom has it that the lower a language is on the configurationality scale, the more morphological marking we expect to be used (Bresnan, 2001, page 6).

For a statistical parser to cope with nonconfigurational phenomena as observed in, for instance, Hebrew or German, it should allow for flexibility in the *form* of realization of the grammatical *functions* within the phrase-structure representation of trees. Recent morphological theories employ *Form-Function* separation as a widely-accepted practice for enhancing the adequacy of models describing variability in the realization of grammatical *properties*. Our results suggest that the adequacy of syntactic processing models is related to such typological insights as well, and is enhanced by adopting a similar form-function separation for expressing grammatical *relations*.

configurational ———————— *nonconfigurational*
Chinese>English>{German,Hebrew}>Warlpiri

Figure 5: **The Configurationality Scale**

The HD assumptions take the function of a constituent to be transparently related to its formal position, which entails word-order rigidity. Such transparent relations between configurational positions and grammatical functions are assumed by other kinds of parsing frameworks such as the 'all-subtrees' approach of Data-Oriented Parsing, and the distinction between left and right application in CCG-based parsers.

The RR modeling strategy stipulates a strict separation between *form* — parametrizing explicitly basic word-order (Greenberg, 1963) and morphological realization (Greenberg, 1954) — and function — parametrizing relational networks borrowed from (Perlmutter, 1982) — which makes it possible to statistically learn complex form-function mapping reflected in the data. This is an adequate means to capture, e.g., morphosyntactic interactions, which characterize the *less-configurational* languages on the scale.

8 Conclusion

In our comparison of the HD and RR modeling approaches, the RR approach is shown to be empirically superior and typologically more adequate for parsing a language exhibiting word-order variation interleaved with extended morphology. HD models are less accurate and more vulnerable to sparseness as they assume transparent mappings between form and function, based on *left* and *right* decompositions hard-wired in the HD representation. RR models, in contrast, employ *form* and *function* separation which allows the statistical model to learn complex correspondance patterns reflected in the data. In the future we plan to investigate how the different models fare against one another in parsing different languages. In particular we wish to examine whether parsing different languages should be pursued by different models, or whether the RR strategy can effectively cope with different languages types. Finally, we wish to explore the implications of RR modeling for applications that consider the form of expression in multiple languages, for instance *Statistical Machine Translation (SMT)*.

9 Acknowledgements

We thank Jelle Zuidema, Inbal Tsarfati, David McCloskey and Yoav Golderg for excellent comments on earlier versions. We also thank Miles Osborne and Tikitu de Jager for comments on the camera-ready draft. All errors are our own. The work of the first author is funded by the Dutch Science Foundation (NWO) grant 017.001.271.

References

J. Aissen. 2003. Differential Object Marking: Iconicity vs. Economy. *Natural Language and Linguistic Theory*, 21.

D. M. Bikel. 2002. Design of a Multi-lingual, Parallel-processing Statistical Parsing Engine. In *Proceedings of HLT*.

J. Bresnan. 2001. *Lexical-Functional Syntax*. Blackwell Textbooks in Linguistics. Blackwell.

E. Charniak. 1996. Tree-Bank Grammars. In *AAAI/IAAI, Vol. 2*.

E. Charniak. 1997. Statistical Parsing with a Context-Free Grammar and Word Statistics. In *AAAI/IAAI*.

M. Collins, J. Hajič, E. Brill, L. Ramshaw, and C. Tillmann. 1999. A Statistical Parser of Czech. In *Proceedings ACL*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

M. Collins. 2003. Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics*.

A. Dubey. 2004. *Statistical Parsing for German: Modeling syntactic properties and annotation differences*. Ph.D. thesis, Saarland University, Germany.

Y. Goldberg and R. Tsarfaty. 2008. A Single Framework for Joint Morphological Segmentation and Syntactic Parsing. In *Proceedings of ACL*.

J.H. Greenberg. 1954. A Quantitative Approach to the Morphological Typology of Language. In R. F. Spencer, editor, *Method and Perspective in Anthropology*. University of Minessota Press.

J. H. Greenberg. 1963. Some Universals of Grammar with Particular Reference to the Order of Meaningful Elements. In Joseph H. Greenberg, editor, *Universals of Language*. MIT Press.

N. Guthmann, Y. Krymolowski, A. Milea, and Y. Winter. 2009. Automatic Annotation of Morpho-Syntactic Dependencies in a Modern Hebrew Treebank. In *Proceedings of TLT*.

K. L. Hale. 1983. Warlpiri and the Grammar of Non-Configurational Languages. *Natural Language and Linguistic Theory*, 1(1).

M. Johnson. 1998. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24(4).

D. Klein and C. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of ACL*.

S. Kubler. 2008. The PaGe Shared task on Parsing German. In *ACL Workshop on Parsing German*.

M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *Proceedings of NEMLAR*.

M. Maamouri, A. Bies, and S. Kulick. 2008. Enhanced Annotation and Parsing of the Arabic treebank. In *Proceedings of INFOS*.

D. M. Magerman. 1995. Statistical Decision-Tree Models for Parsing. In *Proceedings of ACL*.

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*.

P. H. Matthews. 1993. *Morphology*. Cambridge.

D. McClosky, E. Charniak, and M. Johnson. 2008. When is self-training effective for parsing? In *Proceedings of CoLing*.

N. Melnik. 2002. *Verb-Initial Constructions in Modern Hebrew*. Ph.D. thesis, Berkeley, California.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The Shared Task on Dependency Parsing. In *Proceedings of the CoNLL Shared Task*.

D. M. Perlmutter. 1982. Syntactic Representation, Syntactic Levels, and the Notion of a Subject. In Pauline Jacobson and Geoffrey Pullum, editors, *The Nature of Syntactic Representation*. Springer.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of ACL*.

A. Rafferty and C. D. Manning. 2008. Parsing Three German Treebanks: Lexicalized and Unlexicalized Baselines. In *ACL WorkShop on Parsing German*.

R. Scha. 1990. Language Theory and Language Technology; Competence and Performance. In Q. A. M. de Kort and G. L. J. Leerdam, editors, *Computertoepassingen in de Neerlandistiek*. Almere: LVVN.

H. Schmid. 2004. Efficient Parsing of Highly Ambiguous Context-Free Grammars with Bit vectors. In *Proceedings of COLING*.

U. Shlonsky. 1997. *Clause Structure and Word Order in Hebrew and Arabic*. Oxford University Press.

K. Sima'an, A. Itai, Y. Winter, A. Altman, and N. Nativ. 2001. Building a Tree-Bank for Modern Hebrew Text. In *Traitement Automatique des Langues*.

J. J. Song. 2001. *Linguistic Typology: Morphology and Syntax*. Pearson Education Limited, Edinbrugh.

R. Tsarfaty and K. Sima'an. 2008. Relational-Realizational Parsing. In *Proceedings of CoLing*.

R. Tsarfaty. 2006. Integrated Morphological and Syntactic Disambiguation for Modern Hebrew. In *Proceeding of ACL-SRW*.

Enhancement of Lexical Concepts Using Cross-lingual Web Mining

Dmitry Davidov
ICNC
The Hebrew University of Jerusalem
dmitry@alice.nc.huji.ac.il

Ari Rappoport
Institute of Computer Science
The Hebrew University of Jerusalem
arir@cs.huji.ac.il

Abstract

Sets of lexical items sharing a significant aspect of their meaning (*concepts*) are fundamental in linguistics and NLP. Manual concept compilation is labor intensive, error prone and subjective. We present a web-based concept extension algorithm. Given a set of terms specifying a concept in some language, we translate them to a wide range of intermediate languages, disambiguate the translations using web counts, and discover additional concept terms using symmetric patterns. We then translate the discovered terms back into the original language, score them, and extend the original concept by adding back-translations having high scores. We evaluate our method in 3 source languages and 45 intermediate languages, using both human judgments and WordNet. In all cases, our cross-lingual algorithm significantly improves high quality concept extension.

1 Introduction

A *concept* (or lexical category) is a set of lexical items sharing a significant aspect of their meanings (e.g., types of food, tool names, etc). Concepts are fundamental in linguistics and NLP, in thesauri, dictionaries, and various applications such as textual entailment and question answering.

Great efforts have been invested in manual preparation of concept resources such as WordNet (WN). However, manual preparation is labor intensive, which means it is both costly and slow to update. Applications needing data on some very specific domain or on a recent news-related event may find such resources lacking. In addition, manual preparation is error-prone and susceptible to subjective concept membership decisions, frequently resulting in concepts whose terms do not belong to the same level of granularity[1]. As a result, there is a need to find methods for automatic improvement of concept coverage and quality.

The web is a huge up-to-date corpus covering many domains, so using it for concept extension has the potential to address the above problems. The majority of web pages are written in a few salient languages, hence most of the web-based information retrieval studies are done on these languages. However, due to the substantial growth of the multilingual web[2], languages in which concept terms are expressed in the most precise manner frequently do not match the language where information is needed. Moreover, representations of the same concept in different languages may complement each other.

In order to benefit from such cross-lingual information, concept acquisition systems should be able to gather concept terms from many available languages and convert them to the desired language. In this paper we present such an algorithm. Given a set of words specifying a concept in some source language, we translate them to a range of intermediate languages and disambiguate the translations using web counts. Then we discover additional concept terms using symmetric patterns and translate the discovered terms back into the original language. Finally we score the back-translations using their intermediate languages' properties, and extend the original concept by adding back-translations having high scores. The only language-specific resource required by the algorithm are multilingual dictionaries, and its processing times are very modest.

We performed thorough evaluation for 24 concepts in 3 source languages (Hebrew, English and Russian) and 45 intermediate languages. Concept definitions were taken from existing WordNet subtrees, and the obtained new terms were manually

[1] See Section 5.1.1.

[2] http://www.internetworldstats.com/stats7.htm

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 852–861,
Singapore, 6-7 August 2009.

scored by human judges. In all cases we have significantly extended the original concept set with high precision. We have also performed a fully automatic evaluation with 150 concepts, showing that the algorithm can re-discover WN concepts with high precision and recall when given only partial lists as input.

Section 2 discusses related work, Section 3 details the algorithm, Section 4 describes the evaluation protocol and Section 5 presents our results.

2 Related work

One of the main goals of this paper is the extension or automated creation of lexical databases such as WN. Due to the importance of WN for NLP tasks, substantial research was done on direct or indirect automated extension of the English WN (e.g., (Snow et al., 2006)) or WN in other languages (e.g., (Vintar and Fišer, 2008)). The majority of this research was done on extending the tree structure (finding new synsets (Snow et al., 2006) or enriching WN with new relationships (Cuadros and Rigau, 2008)) rather than improving the quality of existing concept/synset nodes. Other related studies develop concept acquisition frameworks for on-demand tasks where concepts are defined by user-provided seeds or patterns (Etzioni et al., 2005; Davidov et al., 2007), or for fully unsupervised database creation where concepts are discovered from scratch (Banko et al., 2007; Davidov and Rappoport, 2006).

Some papers directly target specific applications, and build lexical resources as a side effect. Named Entity Recognition can be viewed as an instance of the concept acquisition problem where the desired concepts contain words that are names of entities of a particular kind, as done in (Freitag, 2004) using co-clustering and in (Etzioni et al., 2005) using predefined pattern types.

The two main algorithmic approaches to the problem are pattern-based concept discovery and clustering of context feature vectors. The latter approach represents word contexts as vectors in some space and uses similarity measures and automatic clustering in that space (Deerwester et al., 1990). Pereira et al.(1993), Curran and Moens (2002) and Lin (1998) use syntactic features in the vector definition. Pantel and Lin (2002) improves on the latter by clustering by committee. Caraballo (1999) uses conjunction and appositive annotations in the vector representation. While great effort has been made for improving the computational complexity of these methods (Gorman and Curran, 2006), they still remain data and computation intensive.

The second major algorithmic approach is to use lexico-syntactic patterns. Patterns have been shown to produce more accurate results than feature vectors, at a lower computational cost on large corpora (Pantel et al., 2004). In concept acquisition, pattern-based methods were shown to outperform LSA by a large margin (Widdows and Dorow, 2002). Since (Hearst, 1992), who used a manually prepared set of initial lexical patterns in order to acquire relationships, numerous pattern-based methods have been proposed for the discovery of concepts from seeds (Pantel et al., 2004; Davidov et al., 2007; Pasca et al., 2006). Most of these studies were done for English, while some show the applicability of their methods to other languages, including Greek, Czech, Slovene and French.

Most of these papers attempt to discover concepts from data available in some specific language. Recently several studies have proposed to utilize a second language or several specified languages in order to extract or extend concepts (Vintar and Fišer, 2008; van der Plas and Tiedemann, 2006) or paraphrases (Bosma and Callison-Burch, 2007). However, these methods usually require the availability of parallel corpora, which limits their usefulness. Most of these methods utilize distributional measures, hence they do not possess the advantages of the pattern-based framework.

Unlike in the majority of recent studies, where the framework is designed with specific languages in mind, in our task, in order to take advantage of information from diverse languages, the algorithm should be able to deal well with a wide variety of possible intermediate languages without any manual adaptations. Relying solely on multilingual dictionaries and the web, our algorithm should be able to discover language-specific patterns and concept terms. While some of the proposed frameworks could potentially be language-independent, little research has been done to confirm this. There are a few obstacles that may hinder applying common pattern-based methods to other languages. Many studies utilize parsing or POS tagging, which frequently depend on the availability and quality of language-specific tools. Some studies specify seed patterns in advance, and

it is not clear whether translated patterns can work well on different languages. Also, the absence of clear word segmentation in some languages (e.g., Chinese) can make many methods inapplicable.

A few recently proposed concept acquisition methods require only a handful of seed words and no pattern pre-specification (Davidov et al., 2007; Pasca and Van Durme, 2008). While these studies avoid some of the obstacles above, it still remains open whether such methods are indeed language-independent. In the translation to intermediate languages part of our framework, we adapt the algorithms in (Davidov and Rappoport, 2006; Davidov et al., 2007) to suit diverse languages (including ones without explicit word segmentation). We also develop a method for efficient automated disambiguation and translation of terms to and from any available intermediate language.

Our study is related to cross-language information retrieval (CLIR/CLEF) frameworks. Both deal with information extracted from a set of languages. However, the majority of CLIR studies pursue different targets. One of the main CLIR goals is the retrieval of *documents* based on explicit queries, when the document language is not the query language (Volk and Buitelaar, 2002). These frameworks usually develop language-specific tools and algorithms including parsers and taggers in order to integrate multilingual *queries* and *documents* (Jagarlamudi and Kumaran, 2007). Our goal is to develop a *language-independent* method using cross-lingual information, for the extension and improvement of *concepts* rather than the retrieval of documents. Besides, unlike in many CLIR frameworks, intermediate languages are not specified in advance and the language of requested data is the same as the language of request, while available information may be found in many different intermediate languages.

3 The Algorithm

Our algorithm is comprised of the following stages: (1) given a set of words in a *source* language as a specification for some concept, we automatically translate them to a diverse set of *intermediate* languages, using multilingual dictionaries; (2) the translations are disambiguated using web counts; (3) for each language, we retrieve a set of web snippets where these translations co-appear and apply a pattern-based concept extension algorithm for discovering additional terms; (4) we translate the discovered terms back to the source language, and disambiguate them; (5) we score the back-translated terms using data on their behavior in the intermediate languages, and merge the sets obtained from different languages into a single one, retaining terms whose score passes a certain threshold. Stages 1-3 of the algorithm have been described in (Davidov and Rappoport, 2009), where the goal was to translate a concept given in one language to other languages. The framework presented here includes the new stages 4-5, and its goal and evaluation methods are completely different.

3.1 Concept specification and translation

We start from a set of words denoting a concept in a given source language. Thus we may use words like *(apple, banana, ...)* as the definition of the concept of fruit or *(bear, wolf, fox, ...)* as the definition of wild animals. In order to reduce noise, we limit the length (in words) of multiword expressions considered as terms. To calculate this limit for a language, we randomly take 100 terms from the appropriate dictionary and set a limit as $Lim_{mwe} = round(avg(length(w)))$ where $length(w)$ is the number of words in term w. For languages like Chinese without inherent word segmentation, $length(w)$ is the number of characters in w. While for many languages $Lim_{mwe} = 1$, some languages like Vietnamese usually require two or more words to express terms.

3.2 Disambiguation of translated terms

One of the problems in utilization of multilingual information is ambiguity of translation. First, in order to apply the concept acquisition algorithm, at least some of the given concept terms must be automatically translated to each intermediate language. In order to avoid reliance on parallel corpora, which do not exist or are extremely small for most of our language pairs, we use bilingual dictionaries. Such dictionaries usually provide many translations, one or more for each sense, so this translation is inherently fuzzy. Second, once we acquire translated term lists for each intermediate language, we need to translate them back to the source language and such back-translations are also fuzzy. In both cases, we need to select the appropriate translation for each term.

While our desire would be to work with as many languages as possible, in practice, some or even

most of the concept terms may be absent from the appropriate dictionary. Such concept terms are ignored.

One way to deal with ambiguity is by applying distributional methods, usually requiring a large single-language corpus or, more frequently, parallel corpora. However, such corpora are not readily available for many languages and domains. Extracting such statistical information on-demand is also computationally demanding, limiting its usability. Hence, we take a simple but effective query-based approach. This approach, while being powerful as we show in the evaluation, only relies on a few web queries and does not rely on any language-specific resources or data.

We use the conjecture that terms of the same concept tend to co-appear more frequently than ones belonging to different concepts[3]. Thus, we select a translation of a term co-appearing most frequently with some translation of a different term of the same concept. We estimate how well translations of different terms are connected to each other. Let $C = \{C_i\}$ be the given seed words for some concept. Let $Tr(C_i, n)$ be the n-th available translation of word C_i and $Cnt(s)$ denote the web count of string s obtained by a search engine. We select a translation $Tr(C_i)$ according to:

$$F(w_1, w_2) = \frac{Cnt(\text{“}w_1 * w_2\text{”}) \times Cnt(\text{“}w_2 * w_1\text{”})}{Cnt(w_1) \times Cnt(w_2)}$$

$$Tr(C_i) = \underset{s_i}{argmax} \left(\max_{\substack{s_j \\ j \neq i}} (F(Tr(C_i, s_i), Tr(C_j, s_j))) \right)$$

We utilize the $Yahoo!$ “x * y”,“x * * y” wildcards that allow to count only co-appearances where x and y are separated by a single word or word pair. As a result, we obtain a set of disambiguated term translations. This method is used both in order to translate from the source language to each intermediate language and to back-translate the newly discovered concept terms from the intermediate to the source language.

The number of queries in this stage depends on the ambiguity of the concept terms' translations. In order to decrease the amount of queries, if there are more than three possible senses we sort them by frequency[4] and take three senses with medium frequency. This allows us to skip the most ambiguous and rare senses without any significant effect on performance. Also, if the number of combinations is still too high (>30), we randomly sample at most 30 of the possible combinations.

3.3 Pattern-based extension of concept terms in intermediate languages

We first mine the web for contexts containing the translations. Then we extract from the retrieved snippets contexts where translated terms co-appear, and detect patterns where they co-appear symmetrically. Then we use the detected patterns to discover additional concept terms. In order to define word boundaries, for each language we manually specify boundary characters such as punctuation/space symbols. This data, along with dictionaries, is the only language-specific data in our framework.

Web mining for translation contexts. In order to get language-specific data, we need to restrict web mining each time to the processed intermediate language. This restriction is straightforward if the alphabet or term translations are language-specific or if the search API supports restriction to this language[5]. In case where there are no such natural restrictions, we attempt to detect and add to our queries a few language-specific frequent words. Using our dictionaries, we find 1–3 of the 15 most frequent words in a desired language that are unique to that language, and we ‘and’ them with the queries to ensure proper language selection. This works well for almost all languages (Esperanto being a notable exception).

For each pair A, B of disambiguated term translations, we construct and execute the following two queries: {“A * B”, “B * A”}[6]. When we have 3 or more terms we also add {A B C D}-like conjunction queries which include 3-5 words. For languages with $Lim_{mwe} > 1$, we also construct queries with several “*” wildcards between terms. For each query we collect snippets containing text fragments of web pages. Such snippets frequently include the search terms. Since $Yahoo!$ $Boss$ allows retrieval of up to the 1000 first results (50 in each query), we collect several thousands snippets. For most of the intermediate languages, only a few dozen queries (40 on the average) are required to obtain sufficient data, and queries can be parallelized. Thus the relevant data can be downloaded

[3] Our results here support this conjecture.

[4] Frequency is estimated by web count for a given word.

[5] Yahoo! allows restriction for 42 languages.

[6] These are Yahoo! queries where enclosing words in “” means searching for an exact phrase and “*” means a wildcard for exactly one arbitrary word.

in seconds. This makes our approach practical for on-demand retrieval or concept verification tasks.

Meta-patterns. Following (Davidov et al., 2007), we seek symmetric patterns to retrieve concept terms. We use two meta-pattern types. First, a *Two-Slot* pattern type constructed as follows:

$$[Prefix]\ C_1\ [Infix]\ C_2\ [Postfix]$$

C_i are slots for concept terms. We allow up to Lim_{mwe} space-separated[7] words to be in a single slot. Infix may contain punctuation, spaces, and up to $Lim_{mwe} \times 4$ words. Prefix and Postfix are limited to contain punctuation characters and/or Lim_{mwe} words.

Terms of the same concept frequently co-appear in lists. To utilize this, we introduce two additional *List* pattern types[8]:

$$[Prefix]\ C_1\ [Infix]\ (C_i\ [Infix])+ \quad (1)$$
$$[Infix]\ (C_i\ [Infix])+\ C_n\ [Postfix] \quad (2)$$

Following (Widdows and Dorow, 2002), we define a pattern graph. Nodes correspond to terms and patterns to edges. If term pair (w_1, w_2) appears in pattern P, we add nodes N_{w_1}, N_{w_2} to the graph and a directed edge $E_P(N_{w_1}, N_{w_2})$ between them.

Symmetric patterns. We consider only symmetric patterns. We define a symmetric pattern as a pattern where some concept terms C_i, C_j appear both in left-to-right and right-to-left order. For example, if we consider the terms $\{apple, pineapple\}$ we select a List pattern "$(one\ C_i,)+\ and\ C_n$." if we find both "one *apple*, one *pineapple*, one guava and orange." and "one watermelon, one *pineapple* and *apple*.". If no such patterns are found, we turn to a weaker definition, considering as symmetric those patterns where the same terms appear in the corpus in at least two different slots. Thus, we select a pattern "for C_1 and C_2" if we see both "for *apple* and guava," and "for orange and *apple*,".

Retrieving concept terms. We collect terms in two stages. First, we obtain "high-quality" core terms and then we retrieve potentially more noisy ones. At the first stage we collect all terms[9] that are bidirectionally connected to at least two different original translations, and call them *core* concept terms C_{core}. We also add the original ones as core terms. Then we detect the rest of the terms C_{rest} that are connected to the core stronger than to the remaining words, as follows:
$G_{in}(c)=\{w \in C_{core} | E(N_w, N_c) \vee E(N_c, N_w)\}$
$G_{out}(c)=\{w \notin C_{core} | E(N_w, N_c) \vee E(N_c, N_w)\}$
$C_{rest}=\{c | |G_{in}(c)| > |G_{out}(c)|\}$

For the sake of simplicity, we do not attempt to discover more patterns/instances iteratively by requerying the web. If we have enough data, we use windowing to improve result quality. If we obtain more than 400 snippets for some concept, we divide the data into equal parts, each containing up to 400 snippets. We apply our algorithm independently to each part and select only the words that appear in more than one part.

3.4 Back-translation and disambiguation

At the concept acquisition phase of our framework we obtained sets of terms for each intermediate language, each set representing a concept. In order to be useful for the enhancement of the original concept, these terms are now back-translated to the source language. We disambiguate each back-translated term using the process described in Section 3.2. Having sets of back-translated terms for each intermediate language, our goal is to combine these into a single set.

3.5 Scoring and merging the back translations

We do this merging using the following scoring strategy, assigning for each proposed term t' in concept C the score $S(t', C)$, and selecting terms with $S(t', C) > H$ where H is a predefined threshold.

Our scoring is based on the two following considerations. First, we assume that terms extracted from more languages tend to be less noisy and language-dependent. Second, we would like to favor languages with less resources for a given concept, since noise empirically appears to be less prominent in such languages[10].

For language L and concept $C = \{t_1 \dots t_k\}$ we get a disambiguated set of translations $\{Tr(t_1, L) \dots Tr(t_k, L)\}$. We define relative lan-

[7] As before, for languages without space-based word separation Lim_{mwe} limits the number of characters instead.

[8] $(E)+$ means one or more instances of E.

[9] We do not consider as terms the 50 most frequent words.

[10] Preliminary experimentation, as well as the evaluation results presented in this paper, support both of these considerations.

guage frequency by

$$LFreq(L,C) = \frac{\sum_{t_i \in C}(Freq(Tr(t_i,L)))}{\sum_{L',t_i \in C}(Freq(Tr(t_i,L'))}$$

where $Freq(Tr(t_i,L))$ is a frequency of term's t_i translation to language L estimated by the number of web hits. Thus languages in which translated concept terms appear more times will get higher relative frequency, potentially indicating a greater concept translation ambiguity. Now, for each new term t' discovered through $LNum(t')$ different languages $L_1 \ldots L_{LNum(t')}$ we calculate a term score [11] $S(t',C)$:

$$S(t',C) = LNum(t') \cdot \left(1 - \sum_i LFreq(L_i,C)\right)$$

For each discovered term t', $S(t',C) \in [0, LNum(t')]$, while discovery of t' in less frequent languages will cause the score to be closer to $LNum(t')$. So terms appearing in a greater number of infrequent languages will get higher scores.

After the calculation of score for each proposed term, we retain terms whose scores are above the predefined threshold H. In our experiments we have used $H = 3$, usually meaning that acquisition of a term through 3-4 uncommon intermediate languages should be enough to accept it. The same score measure can also be used to filter out "bad" terms in an already existing concept.

4 Experimental Setup

We describe here the languages, concepts and dictionaries we used in our experiments.

4.1 Languages and concepts

One of the main goals in this research is to take advantage of concept data in every possible language. As intermediate languages, we used 45 languages including major west European languages like French or German, Slavic languages like Russian, Semitic languages as Hebrew and Arabic, and diverse Asian languages such as Chinese and Persian. To configure parameters we have used a set of 10 concepts in Russian as a development set. These concepts were not used in evaluation.

We examined a wide variety of concepts and for each of them we used all languages with available translations. Table 1 shows the resulting top 10 most utilized languages in our experiments.

[11] In this expression i runs only on languages with term t' hence the summation is not 1.

English	**Russian**	**Hebrew**
German(68%)	English(70%)	English(66%)
French(60%)	German(62%)	German(65%)
Italian(60%)	French(62%)	Italian(61%)
Portuguese(57%)	Spanish(58%)	French(59%)
Spanish(55%)	Italian(56%)	Spanish(57%)
Turkish(51%)	Portuguese(54%)	Portuguese(57%)
Russian(50%)	Korean(50%)	Korean(48%)
Korean(46%)	Turkish(49%)	Russian(43%)
Chinese(45%)	Chinese(47%)	Turkish(43%)
Czech(42%)	Polish (44%)	Czech(40%)

Table 1: The ten most utilized intermediate languages in our experiments. In parentheses we show the percentage of new terms that these languages helped discover.

We have used the English, Hebrew (Ordan and Winter, 2008) and Russian (Gelfenbeynand et al., 2003) WordNets as sources for concepts and for the automatic evaluation. Our concept set selection was based on English WN subtrees. To perform comparable experiments with Russian and Hebrew, we have selected the same subtrees in the Hebrew and Russian WN. Concept definitions given to human judges for evaluation were based on the corresponding WN glosses. For automated evaluation we selected 150 synsets/subtrees containing at least 10 single word terms (existing in all three tested languages).

For manual evaluation we used a subset of 24 of these concepts. In this subset we tried to select generic concepts manually, such that no domain expert knowledge was required to check their correctness. Ten of these concepts were identical to ones used in (Widdows and Dorow, 2002; Davidov and Rappoport, 2006), which allowed us to compare our results to recent work in case of English. Table 2 shows these 10 concepts along with the sample terms. While the number of tested concepts is not very large, it provides a good indication for the quality of our approach.

Concept	Sample terms
Musical instruments	guitar, flute, piano
Vehicles/transport	train, bus, car
Academic subjects	physics, chemistry, psychology
Body parts	hand, leg, shoulder
Food	egg, butter, bread
Clothes	pants, skirt, jacket
Tools	hammer, screwdriver, wrench
Places	park, castle, garden
Crimes	murder, theft, fraud
Diseases	rubella, measles, jaundice

Table 2: Ten of the selected concepts with sample terms.

4.2 Multilingual dictionaries

We developed tools for automatic access to a number of dictionaries. We used Wikipedia cross-language links as our main source ($> 60\%$) for offline translation. These links include translation of Wikipedia terms into dozens of languages. The main advantage of using Wikipedia is its wide coverage of concepts and languages. However, one problem it has is that it frequently encodes too specific senses and misses common ones (*bear* is translated as *family Ursidae*, missing its common "wild animal" sense). To overcome these difficulties, we also used Wiktionary and complemented these offline resources with automated queries to several (25) online dictionaries. We start with Wikipedia definitions, then Wiktionary, and then, if not found, we turn to online dictionaries.

5 Evaluation and Results

Potential applications of our framework include both the extension of existing lexical databases and the construction of new databases from a small set of seeds for each concept. Consequently, in our evaluation we aim to check both the ability to extend nearly complete concepts and the ability to discover most of the concept given a few seeds. Since in our current framework we extend a small subset of concepts rather than the whole database, we could not utilize application-based evaluation strategies such as performance in WSD tasks (Cuadros and Rigau, 2008).

5.1 Human judgment evaluation

In order to check how well we can extend existing concepts, we count and verify the quality of new concept terms discovered by the algorithm given complete concepts from WN. Performing an automatic evaluation of such new terms is a challenging task, since there are no exhaustive term lists available. Thus, in order to check how well newly added terms fit the concept definition, we have to use human judges.

We provided four human subjects with 24 lists of newly discovered terms, together with original concept definitions (written as descriptive natural language sentences) and asked them to rank (1-10, 10 being best) how well each of these terms fits the given definition. We have instructed judges to accept common misspellings and reject words that are too general/narrow for the provided definition.

We mixed the discovered terms with equal amounts of terms from three control sets: (1) terms from the original WN concept; (2) randomly selected WN terms; (3) terms obtained by applying the single-language concept acquisition algorithm described in Section 3.3 in the source language. Kappa inter-annotator agreement scores were above 0.6 for all tests below.

5.1.1 WordNet concept extension

The middle column of Table 3 shows the judge scores and average amount of added terms for each source language. In this case the algorithm was provided with complete term lists as concept definitions, and was requested to extend these lists. We can see that while the scores for original WN terms are not perfect (7/10), single-language and cross-lingual concept extension achieve nearly the same scores. However, the latter discovers many more new concept terms without reducing quality. The difference becomes more substantial for Hebrew, which is a resource-poor source language, heavily affecting the performance of single-language concept extension methods.

The low ranks for WN reflect the ambiguity of definition of some of its classification subtrees. Thus, for the 'body part' concept defined in WordNet as "any part of an organism such as an organ or extremity" (which is not supposed to require domain-specific knowledge to identify) low scores were given (correctly) by judges to generic terms such as tissue, system, apparatus and process (process defined in WN as "a natural prolongation or projection from a part of an organism"), positioned in WN as direct hyponyms of body parts. Low scores were also given to very specific terms like "saddle" (posterior part of the back of a domestic fowl) or very ambiguous terms like "small" (the slender part of the back).

5.1.2 Seed-based concept extension

The rightmost column of Table 3 shows similar information to the middle column, but when only the three most frequent terms from the original WN concept were given as concept definitions. We can see that even given three words as seeds, the cross-lingual framework allows to discover many new terms. Surprisingly, terms extracted by the cross-lingual framework achieve significantly higher scores not only in comparison to the single-language algorithm but also in comparison to existing WN terms. Thus while the "native" WN concept and single-language concept extension re-

sults get a score of 7/10, terms obtained by the cross-lingual framework obtain an average score of nearly 9/10.

This suggests that our cross-lingual framework can lead to better (from a human judgment point of view) assignment of terms to concepts, even in comparison to manual annotation.

	Input	
	all terms	3 terms
English		
WordNet	7.2	7.2
Random	1.8	1.8
SingleLanguage	7.0(10)	7.8(18)
Crosslingual	6.9(19)	8.8(26)
Russian		
WordNet	7.8	7.8
Random	1.9	1.9
SingleLanguage	7.4(10)	8.1(16)
Crosslingual	7.6(21)	9.0(29)
Hebrew		
WordNet	7.0	7.0
Random	1.3	1.3
SingleLanguage	6.5(4)	7.5(6)
Crosslingual	6.8(18)	8.9(24)

Table 3: Human judgment scores for concept extension in three languages (1 . . . 10, 10 is best). The WordNet, Random and SingleLanguage rows provide corresponding baselines. Average count of newly added terms are shown in parentheses. Average original WN concept size in this set was 36 for English, 32 for Russian and 27 for Hebrew.

5.2 WordNet-based evaluation

While human judgment evaluation provides a good indication for the quality of our framework, it has severe limitations. Thus terms in many concepts require domain expertise to be properly labeled. We have complemented human judgment evaluation with automated WN-based evaluation with a greater (150) number of concepts. For each of the 150 concepts, we have applied our framework on a subset of the available terms, and estimated precision and recall of the resulting term list in comparison to the original WN term list. The evaluation protocol and metrics were very similar to (Davidov and Rappoport, 2006; Widdows and Dorow, 2002) which allowed us to do indirect comparison to previous work.

Table 4 shows precision and recall for this task comparing single-language concept extension and the cross-lingual framework. We can see that in all cases, utilization of the latter greatly improves recall. It also significantly outperforms the single-language pattern-based method introduced by (Davidov and Rappoport, 2006), which achieves average precision of 79.3 on a similar set in English (in comparison to 86.7 in this study). We can also see a decrease in precision when the algorithm is provided with 50% of the concept terms as input and had to discover the remaining 50%. However, careful examination of the results shows that this decrease is due to discovery of additional correct terms not present in WordNet.

	Input					
	50% terms			3 terms		
	P	R	F	P	R	F
English						
SingleLanguage	89.2	75.9	82.0	80.6	15.2	25.6
CrossLingual	86.5	91.1	88.7	86.7	60.2	71.1
Russian						
SingleLanguage	91.3	69.0	78.6	82.1	18.3	29.9
CrossLingual	84.9	86.2	85.5	85.3	62.1	71.9
Hebrew						
SingleLanguage	93.8	38.6	54.7	90.2	5.7	10.7
CrossLingual	86.5	82.4	84.4	93.9	55.6	69.8

Table 4: WordNet-based precision (P) and recall (R) for concept extension.

5.3 Contribution of each language

Each of the 45 languages we used influences the score of at least 5% of the discovered terms. However, it is not apparent if all languages are indeed beneficial or if only a handful of languages can be used. In order to check this point we have performed partial automated tests as described in Section 5.2, removing one language at a time. We also tried to remove random subsets of 2-3 languages, comparing them to removal of one of them. We saw that in each case removal of more languages caused a consistent (while sometimes minor) decrease both in precision and recall metrics. Thus, each language contributes to the system.

6 Discussion

We proposed a framework which given a set of terms defining a concept in some language, utilizes multilingual information available on the web in order to extend this list. This method allows to take advantage of web data in many languages, requiring only multilingual dictionaries. Our method was able to discover a substantially greater number of terms than state-of-the-art single language pattern-based concept extension methods, while retaining high precision.

We also showed that concepts obtained by this method tend to be more coherent in comparison to corresponding concepts in WN, a manually prepared resource. Due to its relative language-independence and modest data requirements, this framework allows gathering required

concept information from the web even if it is scattered among different and relatively uncommon or resource-poor languages.

References

Mishele Banko, Michael J Cafarella , Stephen Soderland, Matt Broadhead, Oren Etzioni, 2007. Open information extraction from the Web. *IJCAI '07*.

Wauter Bosma, Chris Callison-Burch, 2007. Paraphrase substitution for recognizing textual entailment.. *Evaluation of Multilingual and Multimodal Information Retrieval, Lecture Notes in Computer Science '07*.

Sharon Caraballo, 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. *ACL '99*.

Montse Cuadros, German Rigau, 2008. KnowNet: Building a large net of knowledge from the Web. *COLING '08*.

James R. Curran, Marc Moens, 2002. Improvements in automatic thesaurus extraction *SIGLEX 02'*, 59–66.

Dmitry Davidov, Ari Rappoport, 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. *COLING-ACL '06.*

Dmitry Davidov, Ari Rappoport, Moshe Koppel, 2007. Fully unsupervised discovery of concept-specific relationships by web mining. *ACL '07*.

Dmitry Davidov, Ari Rappoport, 2009. Translation and extension of concepts across languages. *EACL '09*.

Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, Richard Harshman, 1990. Indexing by latent semantic analysis. *J. of the American Society for Info. Science*, 41(6):391–407.

Beate Dorow, Dominic Widdows, Katarina Ling, Jean-Pierre Eckmann, Danilo Sergi, Elisha Moses, 2005. Using curvature and Markov clustering in graphs for lexical acquisition and word sense discrimination. *MEANING '05*.

Oren Etzioni, Michael Cafarella, Doug Downey, S. Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel Weld, Alexander Yates, 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence, 165(1):91134*.

Dayne Freitag, 2004. Trained named entity recognition using distributional clusters. *EMNLP '04*.

Ilya Gelfenbeyn, Artem Goncharuk, Vladislav Lehelt, Anton Lipatov, Victor Shilo, 2003. Automatic translation of WordNet semantic network to Russian language (in Russian) *International Dialog 2003 Workshop*.

J. Gorman, J.R. Curran, 2006. Scaling distributional similarity to large corpora. *COLING-ACL '06*.

Marti Hearst, 1992. Automatic acquisition of hyponyms from large text corpora. *COLING '92*.

Jagadeesh Jagarlamudi, A Kumaran, 2007. Cross-lingual information retrieval system for Indian languages. *Working Notes for the CLEF 2007 Workshop*.

Dekang Lin, 1998. Automatic retrieval and clustering of similar words. *COLING '98*.

Noam Ordan, Shuly Wintner, 2007. Hebrew WordNet: a test case of aligning lexical databases across languages. *International Journal of Translation 19(1):39-58, 2007*.

Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, Alpa Jain, 2006. Names and similarities on the web: fact extraction in the fast lane. *COLING-ACL '06*.

Marius Pasca, Benjamin Van Durme, 2008. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. *ACL '08*.

Patrick Pantel, Dekang Lin, 2002. Discovering word senses from text. *SIGKDD '02*.

Patrick Pantel, Deepak Ravichandran, Eduard Hovy, 2004. Towards terascale knowledge acquisition. *COLING '04*.

John Paolillo, Daniel Pimienta, Daniel Prado, et al., 2005. Measuring linguistic diversity on the Internet. *UNESCO Institute for Statistics Montreal, Canada*.

Adam Pease, Christiane Fellbaum, Piek Vossen, 2008. Building the global WordNet grid. *CIL18*.

Fernando Pereira, Naftali Tishby, Lillian Lee, 1993. Distributional clustering of English words. *ACL '93*.

Ellen Riloff, Rosie Jones, 1999. Learning dictionaries for information extraction by multi-level bootstrapping. *AAAI '99*.

Rion Snow, Daniel Jurafsky, Andrew Ng, 2006. Semantic taxonomy induction from heterogeneous evidence. *COLING-ACL '06*.

Lonneke van der Plas, Jorg Tiedemann, 2006. Finding synonyms using automatic word alignment and measures of distributional similarity. *COLING-ACL '06*.

Martin Volk, Paul Buitelaar, 2002. A systematic evaluation of concept-based cross-language information retrieval in the medical domain. *In: Proc. of 3rd Dutch-Belgian Information Retrieval Workshop.*

Špela Vintar, Darja Fišer, 2008. Harvesting multiword expressions from parallel corpora. *LREC '08*.

Dominic Widdows, Beate Dorow, 2002. A graph model for unsupervised lexical acquisition. *COLING '02*.

Bilingual dictionary generation for low-resourced language pairs

Varga István
Yamagata University,
Graduate School of Science and Engineering
dyn36150@dip.yz.yamagata-u.ac.jp

Yokoyama Shoichi
Yamagata University,
Graduate School of Science and Engineering
yokoyama@yz.yamagata-u.ac.jp

Abstract

Bilingual dictionaries are vital resources in many areas of natural language processing. Numerous methods of machine translation require bilingual dictionaries with large coverage, but less-frequent language pairs rarely have any digitalized resources. Since the need for these resources is increasing, but the human resources are scarce for less represented languages, efficient automatized methods are needed. This paper introduces a fully automated, robust pivot language based bilingual dictionary generation method that uses the WordNet of the pivot language to build a new bilingual dictionary. We propose the usage of WordNet in order to increase accuracy; we also introduce a bidirectional selection method with a flexible threshold to maximize recall. Our evaluations showed 79% accuracy and 51% weighted recall, outperforming representative pivot language based methods. A dictionary generated with this method will still need manual post-editing, but the improved recall and precision decrease the work of human correctors.

1 Introduction

In recent decades automatic and semi-automatic machine translation systems gradually managed to take over costly human tasks. This much welcomed change can be attributed not only to major developments in techniques regarding translation methods, but also to important translation resources, such as monolingual or bilingual dictionaries and corpora, thesauri, and so on. However, while widely used language pairs can fully take advantage of state-of-the-art developments in machine translation, certain low-frequency, or less common language pairs lack some or even most of the above mentioned translation resources. In that case, the key to a highly accurate machine translation system switches from the choice and adaptation of the translation method to the problem of available translation resources between the chosen languages.

One possible solution is bilingual corpus acquisition for statistical machine translation (SMT). However, for highly accurate SMT systems large bilingual corpora are required, which are rarely available for less represented languages. Rule or sentence pattern based systems are an attractive alternative, for these systems the need for a bilingual dictionary is essential.

Our paper targets bilingual dictionary generation, a resource which can be used within the frameworks of a rule or pattern based machine translation system. Our goal is to provide a low-cost, robust and accurate dictionary generation method. Low cost and robustness are essential in order to be re-implementable with any arbitrary language pair. We also believe that besides high precision, high recall is also crucial in order to facilitate post-editing which has to be performed by human correctors. For improved precision, we propose the usage of WordNet, while for good recall we introduce a bidirectional selection method with local thresholds.

Our paper is structured as follows: first we overview the most significant related works, after which we analyze the problems of current dictionary generation methods. We present the details of our proposal, exemplified with the Japanese-Hungarian language pair. We evaluate the generated dictionary, performing also a comparative evaluation with two other pivot-language based methods. Finally we present our conclusions.

2 Related works

2.1 Bilingual dictionary generation

Various corpus based, statistical methods with very good recall and precision were developed starting from the 1980's, most notably using the

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 862–870,
Singapore, 6-7 August 2009.

Dice-coefficient (Kay & Röscheisen, 1993), *correspondence-tables* (Brown, 1997), or *mutual information* (Brown et al., 1998).

As an answer to the corpus-based method's biggest disadvantage, namely the need for a large bilingual corpus, in the 1990's Tanaka and Umemura (1994) presented a new approach. As a resource, they only use dictionaries to and from a pivot language to generate a new dictionary. These so-called pivot language based methods rely on the idea that the lookup of a word in an uncommon language through a third, intermediated language can be automated. Tanaka and Umemura's method uses bidirectional source-pivot and pivot-target dictionaries (harmonized dictionaries). Correct translation pairs are selected by means of inverse consultation, a method that relies on counting the number of pivot language definitions of the source word, through which the target language definitions can be identified (Tanaka and Umemura, 1994).

Sjöbergh (2005) also presented an approach to pivot language based dictionary generation. When generating his English pivoted Swedish-Japanese dictionary, each Japanese-to-English description is compared with each Swedish-to-English description. Scoring is based on word overlap, weighted with inverse document frequency; the best matches being selected as translation pairs.

These two approaches described above are the best performing ones that are general enough to be applicable with other language pairs as well. In our research we used these two methods as baselines for comparative evaluation.

There are numerous refinements of the above methods, but for various reasons they cannot be implemented with any arbitrary language pair. Shirai and Yamamoto (2001) used English to design a Korean-Japanese dictionary, but because the usage of language-specific information, they conclude that their method *'can be considered to be applicable to cases of generating among languages similar to Japanese or Korean through English'*. In other cases, only a small portion of the lexical inventory of the language is chosen to be translated: Paik et al. (2001) proposed a method with multiple pivots (English and Kanji/Hanzi characters) to translate Sino-Korean entries. Bond and Ogura describe a Japanese-Malay dictionary that uses a novel technique in its improved matching through normalization of the pivot language, by means of semantic classes, but only for nouns (2007). Besides English, they also use Chinese as a second pivot.

2.2 Lexical database in lexical acquisition

Large lexical databases are vital for many areas in natural language processing (NLP), where large amount of structured linguistic data is needed. The appearance of WordNet (Miller et al., 1990) had a big impact in NLP, since not only did it provide one of the first wide-range collections of linguistic data in electronic format, but it also offered a relatively simple structure that can be implemented with other languages as well. In the last decades since the first, English WordNet, numerous languages adopted the WordNet structure, thus creating a potential large multilingual network. The Japanese language is one of the most recent ones added to the WordNet family (Isahara et al. 2008), but the Hungarian WordNet is still under development (Prószéky et al. 2001; Miháltz and Prószéky 2004).

Multilingual projects, such as EuroWordNet (Vossen 1998; Peters et al. 1998), Balkanet (Stamou et al. 2002) or Multilingual Central Repository (Agirre et al. 2007) aim to solve numerous problems in natural language processing. EuroWordNet was specifically designed for word disambiguation purposes in cross-language information retrieval (Vossen 1998). The internal structure of the multilingual WordNets itself can be a good starting point for bilingual dictionary generation. In case of EuroWordNet, besides the internal design of the initial WordNet for each language, an Inter-Lingual-Index interlinks word meaning across languages is implemented (Peters et al. 1998). However, there are two limitations: first of all, the size of each individual language database is relatively small (Vossen 1998), covering only the most frequent words in each language, thus not being sufficient for creating a dictionary with a large coverage. Secondly, these multilingual databases cover only a handful of languages, with Hungarian or Japanese not being part of them. Adding a new language would require the existence of a WordNet of that language.

3 Problems of current pivot language based methods

3.1 Selection method shortcomings

Previous pivot language based methods generate and score a number of translation candidates, and the candidate's scores that exceed a certain predefined global threshold are selected as viable translation pairs. However, the scores highly de-

pend on the entry itself or the number of translations in the pivot language, therefore there is a variance in what that score represents. For this reason, a large number of good entries are entirely left out from the dictionary, because all of their translation candidates scored low, while faulty translation candidates are selected, because they exceed the global threshold. Due to this effect the recall value drops significantly.

3.2 Dictionaries not enough as resource

Regardless of the language pair, in most cases the meanings of the corresponding words are not identical; they only overlap to a certain extent. Therefore, the pivot language based dictionary generation problem can be defined as the identification of the common elements or the extent of the relevant overlapping in the source-to-pivot and target-to-pivot definitions.

Current methods perform a strictly lexical overlap of the source-pivot and target-pivot entries. Even if the meanings of the source and target head words are transferred to the pivot language, this is rarely done with the same set of words or definitions. Thus, due to the different word-usage or paraphrases, even semantically identical or very similar head words can have different definitions in different dictionaries. As a result, performing only lexical overlap, current methods cannot identify the differences between totally different definitions resulted by unrelated concepts, and differences in only nuances resulted by lexicographers describing the same concept, but with different words.

4 Proposed method

4.1 Specifics of our proposal

For higher precision, instead of the familiar lexical overlap of the current methods we calculate the semantically expanded lexical overlap of the source-to-pivot and target-to-pivot translations. In order to do that, we use semantic information extracted from the WordNet of the pivot language.

To improve recall, we introduce *bidirectional selection.* As we stated above, the global threshold eliminates a large number of good translation pairs, resulting in a low recall. As a solution, we can group the translations that share the same source or target entry, and set *local thresholds* for each head word. For example, for a source language head word *entry_source* there could be multiple target language candidates: $entry\_target_1, \ldots, entry\_target_n$. If the top scoring $entry\_target_k$ candidates are selected, we ensure that at least one translation will be available for *entry_source*, maintaining a high recall. Since we can group the entries in the source language and target language as well, we perform this selection twice, once in each direction. Local thresholds depend on the top scoring *entry_target*, being set to $maxscore \cdot c$. Constant c varies between 0 and 1, allowing a small window not only for the maximum, but high scoring candidates as well. It is language and selection method dependent (see §5.1 for details).

4.2 Translation resources

As an example of a less-common language pair, we have chosen Japanese and Hungarian. For translation candidate generation, we have chosen two freely available dictionaries with English as the pivot language. The Japanese-English dictionary had 197282, while the Hungarian-English contained 189331 1-to-1 entry pairs. The Japanese-English dictionary had part-of-speech (POS) information as well, but to ensure robustness, our method does not use this information.

To select from the translation candidates, we mainly use *WordNet* (Miller et. al., 1990). From WordNet we consider four types of information: *sense categorization, synonymy*, *antonymy* and *semantic categories* provided by the tree structure of nouns and verbs.

4.3 Dictionary generation method

Our proposed method consists of two steps. In step 1 we generate a number of translation pair candidates, while in step 2 we score and select from them based on semantic information extracted from WordNet.

Step 1: translation candidate generation

Using the source-pivot and pivot-target dictionaries, we connect the source and target entries that share at least one common translation in the pivot language. We consider each source-target pair a *translation candidate*. With our Japanese-English and English-Hungarian dictionaries we accumulated 436966 Japanese-Hungarian translation candidates.

Step 2: translation pair selection

We examine the translation candidates one by one, looking up the source-pivot and target-pivot dictionaries, comparing the translations in the pivot language. There are six types of translations that we label *A-F* and explain below. First,

we perform a strictly lexical match based only on the dictionaries. Next, using information extracted from WordNet we attempt to identify the correct translation pairs.

(a) Lexically unambiguous translation pairs

Some of the translation candidates have exactly the same translations into in the pivot language; we consider these pairs as being correct by default. Also among the translation candidates we identified a number of source entries that had only one target translation; and a number of target entries that had only one source translation. Being the sole candidates for the given entries, we consider these pairs too as being correct. 37391 Japanese-Hungarian translation pairs were retrieved with this method (*type A* pairs).

(b) Using sense description

For most polysemous words WordNet has detailed descriptions with synonyms for each sense. We use these synonyms of WordNet's sense descriptions to disambiguate the meanings of the common translations. For a given source-target translation candidate (s,t) we look up the source-pivot and target-pivot translations ($s \rightarrow I = \{s \rightarrow i_1, \ldots, s \rightarrow i_n\}$ and $t \rightarrow I = \{t \rightarrow i_1, \ldots, t \rightarrow i_m\}$). We select the elements that are common in the two definitions ($I' = (s \rightarrow I) \cap (t \rightarrow I)$) and we look up their respective senses from WordNet ($sns(I')$). We identify the words' senses comparing each synonym in the WordNet's synonym description with each word from the dictionary definition. As a result, for each common word we arrive at a certain set of senses from the source-pivot definitions ($sns((s \rightarrow I')$) and a certain set of senses from the target-pivot definitions ($sns((t \rightarrow I')$). We mark $score_B(s,t)$ the maximum ratio of the identical and total number of identified senses (Jaccard coefficient). The higher the $score_B(s,t)$ is, the more probable is candidate (s,t) a valid translation.

$$score_B(s,t) = \max_{i' \in (s \rightarrow I) \cap (t \rightarrow I)} \frac{|sns(s \rightarrow i') \cap sns(t \rightarrow i')|}{|sns(s \rightarrow i') \cup sns(t \rightarrow i')|} \quad (1)$$

For example, *正解 (seikai: correct, right, correct interpretation)* and *helyes (correct, proper, right, appropriate)* have two common translations ($I' = \{right, correct\}$), thus $score_B(s,t)$ can be performed with these two words. The adjective *right* has 13 senses according to WordNet, among them 4 were identified from the Japanese to English definition ($sns(right) = \{\#1, \#3, \#5, \#10\}$, all identified through *correct*) and 5 from the Hungarian to English definition ($sns(right) = \{\#1, \#3, \#5, \#6, \#10\}$, through *correct* or *proper*). As a result, 4 senses are common, and 1 is different. Thus the adjective *right*'s score is 0.8 ($score_B(s,t)[right](正解, helyes)$). The adjective *correct* has 4 senses, all of them are recognized by both definitions through *right*, therefore the score through *correct* is 1 ($score_B(s,t)[correct](正解, helyes)$). The maximum of the above scores is the final score: $score_B(s,t)(正解, helyes) = 1$.

All translation candidates are verified based on all four POS available from WordNet. Since synonymy information is available for nouns (N), verbs (V), adjectives (A) and adverbs (R), four separate scores are calculated for each POS.

Scores that pass a global threshold are considered correct. 33971 Japanese-Hungarian candidates (*type B* translations) were selected, with these two languages the global threshold was set to 0.1. Even this low value ensures that at least one of ten meanings is shared by the two entries of the pair, thus being suitable as translation pair.

(c) Using synonymy, antonymy and semantic categories

We expand the source-to-pivot and target-to-pivot definitions with information from WordNet (synonymy, antonymy and semantic category, respectively). Thus the similarity of the two expanded pivot language descriptions gives a better indication on the suitability of the translation candidate. Using the three relations, the common versus total number of translations (Jaccard coefficient) will define the appropriateness of the translation candidate.

$$score_{C,D,E}(s,t) = \frac{|ext(s \rightarrow i) \cap ext(t \rightarrow i)|}{|ext(s \rightarrow i) \cup ext(t \rightarrow i)|} \quad (2)$$

Since the same word or concept's translations into the pivot language also share the same semantic value, the extension with synonyms ($ext(l \rightarrow i) = (l \rightarrow i) \cup syn(l \rightarrow i)$, where $l = \{s,t\}$) the extended translation should share more common elements.

In case of antonymy, we expand the initial definitions with the *antonyms of the antonyms* ($ext(l \rightarrow i) = (l \rightarrow i) \cup ant(ant(l \rightarrow i))$, where $l = \{s,t\}$). This extension is different from the synonymy extension, in most cases the resulting set of words being considerably larger.

Along with synonymy, antonymy is also available for nouns, verbs, adjectives and adverbs, four separate scores are calculated for each POS.

Semantic categories are provided by the tree structure (hypernymy/hyponymy) of nouns and verbs of WordNet. We transpose each entry from the pivot translations to its semantic categories ($ext(l \rightarrow i) = \Sigma semcat(l \rightarrow i)$, where $l = \{s,t\}$). We assume that the correct translation pairs share a high percentage of semantic categories. Accordingly, the translations of semantically similar or identical entries should share a high number of common semantic categories.

The scores based on these relations highly depend on the number of pivot language translations; therefore we use the bidirectional selection method with local thresholds for each source and target head word. Local thresholds are set based on the best scoring candidate for a given entry. The thresholds were *maxscore*·0.9 for synonymy and antonymy; and *maxscore*·0.8 for the semantic categories (see §5.1 for details).

Using synonymy, 196775 candidate pairs (*type C*), with antonymy 99614 pairs (*type D*); while with semantic categories 195480 pairs (*type E*) were selected.

(d) Combined semantic information

The three separate lists of type C, D and E selection methods resulted in slightly different results, proving that they cannot be used as standalone selection methods (see §5.2 for details).

Because of the multiple POS labelling of numerous words in WordNet, many translation pairs can be selected up to four times based on separate POS information (noun, verb, adjective, adverb), all within one single semantic information based methods. Since we use a bidirectional selection method, experiments showed that translation pairs that were selected during both directions, in most cases were the correct translations. Similarly, translation pairs selected during only one direction were less accurate. In other words, translation pairs whose target language translation was selected as a good translation for the source language entry; and whose source language translation was also selected as a good translation for the target language entry, should be awarded with a higher score. In the same way, entries selected only during one direction should receive a penalty. For every translation candidate we select the maximum score from the several POS (noun, verb, adjective and adverb for synonymy and antonymy relations; noun and verb for semantic category) based scores, multiplied by a multiplication factor (*mfactor*). The multiplication factor varies between 0 and 1, awarding the candidates that were selected both times during the double directional selection; and punishing when selection was made only in a single direction. The product gives the combined score ($score_F$), c_1, c_2 and c_3 are constants. In case of Japanese and Hungarian, these method scored best with the constants set to 1, 0.5 and 0.8, respectively. The combined score also highly depends on the word entry, therefore local thresholds are used in this selection method as well, which were empirically set to *maxscore·0.85* (see §5.1 for details).

$$score_F(s,t) = \prod_{rel} \begin{pmatrix} (c_1 + \max(score_{rel}(s,t))) \cdot \\ (c_2 + c_3 \cdot mfactor_{rel}(s,t)) \end{pmatrix} \qquad (3)$$

As an example, for the Japanese entry 購入 *(kōnyū: buy, purchase)* there are 10 possible Hungarian translations; using the above methods 5 of them (#1, #7, #8, #9, #10) are selected as correct ones. Among these, only 1 of them (#1) is a correct translation, the rest have similar or totally different meanings. However, with the combined scores the faulty translations were eliminated and a new, correct, but previously average scoring translation (#2) was selected (Table 1).

#	translation candidate	$score_F$	$score_C$				$score_D$				$score_E$	
			N	V	A	R	N	V	A	R	N	V
1	vétel *(purchase)*	**2.012**	0.193	0.096	0	0	0	**0.500**	0	0	0.154	**0.500**
2	üzlet *(business transaction)*	**1.387**	0.026	0.030	0	0	0	0.250	0	0	0.020	0.077
3	hozam *(output, yield)*	1.348	0.095	0.071	0	0	0	0	0	0	0.231	0.062
4	emelőrúd *(lever, purchase)*	1.200	0.052	0.079	0	0	0	0	0	0	0.111	0.067
5	előny *(advantage, virtue)*	1.078	0.021	0.020	0	0	0	0	0	0	0.054	0.056
6	támasz *(purchase, support)*	1.053	0.014	0.015	0	0	0	0	0	0	0.037	0.031
7	vásárlás *(shopping)*	0.818	0.153	**0.285**	0	0	0	0	0	0	0.273	0.200
8	szerzemény *(attainment)*	0.771	0.071	**0.285**	0	0	0	0	0	0	0.136	0.200
9	könnyítés *(facilitation)*	0.771	0.064	**0.285**	0	0	0	0	0	0	0.136	0.200
10	emelőszerkezet *(lever)*	0.459	**0.285**	**0.285**	0	0	0	0	0	0	**0.429**	0.200

Table 1: Translation candidate scoring for 購入: *buy, purchase* (above thresholds in bold)

161202 translation pairs were retrieved with this method (*type F*).

During pre-evaluation *type A* and *type B* translations received a score of above 75%, while *type C*, *type D* and *type E* scored low (see §5.2 for details). However, *type F* translations scored close to 80%, therefore from the six translation methods presented above we chose only three (*type A*, *B* and *F*) to construct the dictionary, while the remaining three methods (*type C*, *D* and *E*) are used only indirectly for *type F* selection.

With the described selection methods 187761 translation pairs, with 48973 Japanese and 44664 Hungarian unique entries was generated.

5 Threshold settings and pre-evaluation

5.1 Local threshold settings

As development set we considered all translation candidates whose Hungarian entry starts with "zs" (IPA: ʒ). We assume that the behaviour of this subset of words reflects the behaviour of the entire vocabulary. 133 unique entries totalling 515 translation candidates comprise this development set. After this, we manually scored the 515 translation candidates as *correct* (the translation conveys the same meaning, or the meanings are slightly different, but in a certain context the translation is possible) or *wrong* (the translation pair's two entries convey a different meaning). The scoring was performed by one of the authors who is a native Hungarian and fluent in Japanese. 273 entries were marked as *correct*. Next, we experimented with a number of thresholds to determine which ones provide with the best F-scores (Table 2). The F-scores were determined as follows: for example using synonymy information (type C) in case of threshold=0.85%, 343 of the 515 translation pairs were above the threshold. Among these, 221 were marked as correct by our manual evaluator, thus the precision being 221/343·100=64.43 and the recall being 221/273·100=80.95. F-score is the harmonic mean of precision and recall (71.75 in this case).

selection type	**threshold value** (%)				
	0.75	**0.80**	**0.85**	**0.90**	**0.95**
C	70.27	70.86	71.75	**72.81**	66.95
D	69.92	70.30	70.32	**70.69**	66.66
E	73.71	**74.90**	72.52	71.62	65.09
F	78.78	79.07	**79.34**	78.50	76.94

Table 2: Selection type F-scores with varying thresholds (best threshold values in bold)

5.2 Selection method evaluation

As a pre-evaluation of the above selection methods, we randomly selected 200 1-to-1 source-target entries resulted by each method. The same evaluator scored the translation pairs as *correct* (the translation conveys the same meaning, or the meanings are slightly different, but in a certain context the translation is possible), *undecided* (the translation pair's semantic value is similar, but a translation based on them would be faulty) or *wrong* (the translation pair's two entries convey a different meaning).

selection type	**evaluation score** (%)		
	correct	*undecided*	*wrong*
A	75.5	6.5	18
B	83	7	10
C	68	5.5	26.5
D	60	9	31
E	71	5.5	23.5
F	79	5	16

Table 3: Selection type evaluation

The results showed that *type A* and *type B* selections scored higher than all order-based selections, with *type C*, *type D* and *type E* selections failing to deliver the desired accuracy (Table 3).

6 Evaluation

We performed three types of evaluation:
(1) frequency-weighted recall evaluation
(2) 1-to-1 entry precision evaluation
(3) 1-to-multiple entry evaluation

For comparative purposes we also performed each type of evaluation for two other pivot language based methods whose characteristics permit to be implementable with virtually any language pair. In order to do so, we constructed two other Hungarian-Japanese dictionaries using the methods proposed by Tanaka & Umemura and Sjöbergh, using the same source dictionaries.

6.1 Recall evaluation

It is well known that one of the most challenging aspects of dictionary generation is word ambiguity. It is relatively easy to automatically generate the translations of low-frequency keywords, because they tend to be less ambiguous. On the contrary, the ambiguity of the high frequency words is much higher than their low-frequency counterparts, and as a result conventional methods fail to translate a considerable number of them. However, this discrepancy is not reflected in the traditional recall evaluation, since each

word has an equal weight, regardless of its frequency of use. As a result, we performed a frequency weighted recall evaluation. We used a Japanese frequency dictionary (F_D) generated from the Japanese EDR corpus (Isahara, 2007) to weight each Japanese entry. Setting the standard to the frequency dictionary (its recall value being 100), we automatically search for each entry (w) from the frequency dictionary, looking whether or not it is included in the bilingual dictionary (W_D). If it is recalled, we weight it with its frequency from the frequency dictionary.

$$recall_w = \frac{\sum_{w \in W_D} frequency(w)}{\sum_{w \in F_D} frequency(w)} \cdot 100 \quad (4)$$

method	recall
our method	51.68
Sjöbergh method	37.03
Tanaka method	30.76
initial candidates	51.68
Japanese-English(*)	73.23

Table 4: Recall evaluation results (* marks a manually created dictionary)

The frequency weighted recall value results show that our method's dictionary (51.68) outscores every other automatically generated method's dictionary (37.03, 30.76) with a significant advantage. Moreover, it maintains the score of the initial translation candidates, therefore managing to maximize the recall value, owing to the bidirectional selection method with local thresholds. However, the recall value of a manually created Japanese-English dictionary is higher than any automatically generated dictionary's value (Table 4).

6.2 1-to-1 precision evaluation

With 1-to-1 precision evaluation we determine the translation accuracy of our method, compared with the two baseline methods. 200 random pairs were selected from each of the three Hungarian-Japanese dictionaries, scoring them manually the same way as with selection type evaluation (*correct*, *undecided*, *wrong*) (Table 5). The manual scoring was performed by one of the authors, who is a native Hungarian and fluent in Japanese. Since no independent evaluator was available for these two languages, after a random identification code being assigned to each of the 600 selected translation pairs (200 from each dictionary), they were mixed. Therefore the evaluator did not know the origin of the translation pairs, only after manual scoring the total score for each dictionary was available, after regrouping based on the initial identification codes. The process was repeated 10 times, 2000 pairs were manually checked from each dictionary.

code	Japanese entry	Hungarian entry	classification
k9g6 n5d8	報告 (hōkoku: information, report)	hír (report, information, news)	*correct*
j8h0 k1x5	初 (ubu: innocent, naive)	zöld (green, verdant)	*undecided*
a5b6 n8i3	エントリ (entori: entry <a contest>)	bejárat (entry, entrance)	*wrong*

Table 5: 1-to-1 precision evaluation examples

method	evaluation score (%)		
	correct	*undecided*	*wrong*
our method	79.15%	6.15%	14.70%
Sjöbergh method	54.05%	9.80%	36.15%
Tanaka method	62.50%	7.95%	29.55%

Table 6: 1-to-1 precision evaluation results

To rank the methods we only consider the *correct* translations. Our method performed best with an average of 79.15%, outscoring Tanaka method's 62.50% and Sjöbergh method's 54.05% (Table 6). The maximum deviance of the *correct* translations during the 10 repetitions was less than 3% from the average.

6.3 1-to-multiple evaluation

While with 1-to-1 precision evaluation we estimated the accuracy of the translation pairs, with 1-to-multiple we calculate the true reliability of the dictionary, with the initial translation candidates set as recall benchmark. When looking up the meanings or translations of a certain head word, the user, whether he's a human or a machine, expects all translations to be accurate. Therefore we evaluated 200 randomly selected Japanese entries from the initial translation candidates, together with all of their Hungarian translations, scoring them as *correct* (all translations are correct), *acceptable* (the good translations are predominant, but there are up to 2 erroneous translations), *wrong* (the number or wrong translations exceeds 2) or *missing* (the translation is missing) (Table 7).

The same type of mixed, manual evaluation was performed by the same author on samples of 200 entries from each Japanese-Hungarian dictionary. This evaluation was also repeated 10 times.

To rank the methods, we only consider the *correct* translations. Our method scored best with

71.45%, outperforming Sjöbergh method's 61.65% and Tanaka method's 46.95% (Table 8).

code	Japanese entry	Hungarian translations	classification
j4h8 m9x 5	圧縮 (asshuku: compression, squeeze)	összenyomás (compression, crush, squeeze: *correct*) összeszorítás (compression, confinement: *correct*) zsugorítás (shrinkage: *correct*)	*correct*
h9j9l 3v1	底面 (teimen: base)	alap (base, bottom, foundation: *correct*) alapzat (base, bed, bottom: *correct*) lúg (alkali, base: *undecided*) támpont (base: *correct*)	*acceptable*
l0k6 m3n 7	鳴らす (narasu: to sound, to ring, to beat)	bekerít (to encircle, to enclose, to ring: *wrong*) cseng (to clang, to clank, to ring, to tinkle: *correct*) hangzik (to ring, to sound: *correct*) horkan (to snort: *wrong*) üt (to bang, to knock, to ring: *wrong*)	*wrong*

Table 7: 1-to-multiple entry evaluation examples

method	evaluation score (%)			
	correct	*acceptable*	*wrong*	*missing*
our method	71.45	13.85	14.70	0
Sjöbergh method	61.65	11.30	15.00	12.05
Tanaka method	46.95	3.35	9.10	40.60

Table 8: 1-to-many evaluation results

7 Discussion

Based on the recall evaluations, the traditional methods showed their major weakness by losing substantially from the initial recall values, scored by the initial translation candidates. Our method maintains the same value with the translation candidates, but we cannot say that the recall is perfect. When compared with a manually created dictionary, our method also lost significantly.

Precision evaluation also showed an improvement compared with the traditional methods, our method outscoring the other two methods with the 1-to-1 precision evaluation. 1-to-multiple evaluation was also the highest, proving that WordNet based methods outperform dictionary based methods. Discussing the weaknesses of our system, we have to divide the problems into two categories: recall problems deal with the difficulty in connecting the target and source entries through the pivot language, while precision problems discuss the reasons why erroneous pairs are produced.

7.1 Recall problems

We managed to maximize the recall of our initial translation candidates, but in many cases certain translation pairs still could not be generated because the link from the source language to the target language through the pivot language simply doesn't exist. The main reasons are: the entry is missing from at least one of the dictionaries; translations in the pivot language are expressions or explanations; or there is no direct translation or link between the source and target entries. The entries that could not be recalled are mostly expressions, rare entries, words specific to a language (ex: *tatami: floor-mat,* or *gulyás: goulash*).

Moreover, a number of head words don't have any synonym, antonym and/or hypernymy/hyponymy information in WordNet, and as a result these words could not participate in the type B, C, D, E and F scoring.

7.2 Precision problems

We identified two types of precision problems. The most obvious reasons for erroneous translations are the polysemous nature of words and the meaning-range differences across languages. With words whose senses are clear and mostly preserved even through the pivot language, most of the correct senses were identified and correctly translated. Nouns, adjectives and adverbs had a relatively high degree of accuracy. However, verbs proved to be the most difficult POS to handle. Because semantically they are more flexible than other POS categories, and the meaning range is also highly flexible across languages, the identification of the correct translation is increasingly difficult. For this reason, the number of faulty translations and the number of meanings that are not translated was relatively high.

One other source of erroneous translations is the quality of the initial dictionaries. Even the unambiguous *type A* translations fail to produce the desired accuracy, although they are the unique candidate for a given word entry. The main reason for this is the deficiency of the initial dictionaries, which contain a great number of irrelevant or low usage translations, shadowing the main, important senses of some words. In other cases the resource dictionaries don't contain translations of all meanings; homonyms are

present as pivot entries with different meanings, sometimes creating unique, but faulty links.

8 Conclusions

We proposed a new pivot language based method to create bilingual dictionaries that can be used as translation resource for machine translation. In contrast to conventional methods that use dictionaries only, our method uses WordNet as a main resource of the pivot language to select the suitable translation pairs. As a result, we eliminate most of the weaknesses caused by the structural differences of dictionaries, while profiting from the semantic relations provided by WordNet. We believe that because of the nature of our method it can be re-implemented with most language pairs.

In addition, owing to features such as the bidirectional selection method with local thresholds we managed to maximize recall, while maintaining a precision which is better than any other compared method's score. During exemplification, we generated a mid-large sized Japanese-Hungarian dictionary with relatively good recall and promising precision.

The dictionary is freely available online (http://mj-nlp.homeip.net/mjszotar), being also downloadable at request.

References

Agirre, E., Alegria, I., Rigau, G, Vossen, P. 2007. MCR for CLIR, *Procesamiento del lenguaje natural* 38, pp 3-15.

Bond, F., Ogura, K. 2007. Combining linguistic resources to create a machine-tractable Japanese-Malay dictionary, *Language Resources and Evaluation*, 42(2), pp. 127-136.

Breen, J.W. 1995. Building an Electric Japanese-English Dictionary, *Japanese Studies Association of Australia Conference*, Brisbane, Queensland, Australia.

Brown, P., Cocke, J., Della Pietra, S., Della Pietra, V., Jelinek, F., Mercer, R., Roossin, P. 1998. A Statistical Approach to Language Translation, *Proceedings of COLING-88*, pp. 71-76.

Brown, R.D. 1997. Automated Dictionary Extraction for Knowledge-Free Example-Based Translation, *Proceedings of the 7th International Conference on Theoretical and Methodological Issues in Machine Translation*, pp. 111-118.

Isahara, H., Bond, F., Uchimoto, K., Uchiyama, M., Kanzaki, K. 2008. Development of Japanese WordNet, *Proceedings of LREC-2008*.

Isahara, H. 2007. EDR Electronic Dictionary – present status (EDR 電子化辞書の現状), *NICT-EDR symposium*, pp. 1-14. (in Japanese)

Kay, M., Röscheisen, M. 1993. Text-Translation Alignment, *Computational Linguistics*, 19(1), pp. 121-142.

Miháltz, M., Prószéky, G. 2004. Results and Evaluation of Hungarian Nominal WordNet v1.0, *Proceedings of the Second Global WordNet Conference*, pp. 175-180.

Miller G.A., Beckwith R., Fellbaum C., Gross D., Miller K.J. (1990). Introduction to WordNet: An Online Lexical Database, *Int J Lexicography* 3(4), pp. 235-244.

Paik, K., Bond, F., Shirai, S. 2001. Using Multiple Pivots to align Korean and Japanese Lexical Resources, *NLPRS-2001*, pp. 63-70, Tokyo, Japan.

Peters, W., Vossen, P., Díez-Orzas, P., Adriaens, G. 1998. Cross-linguistic Alignment of Wordnets with an Inter-Lingual-Index, *Computers and the Humanities* 32, pp. 221–251.

Prószéky, G., Miháltz, M., Nagy, D. 2001. Toward a Hungarian WordNet, *Proceedings of the NAACL 2001 Workshop on WordNet and Other Lexical Resources*, Pittsburgh, June 2001.

Sjöbergh, J. 2005. Creating a free Japanese-English lexicon, *Proceedings of PACLING*, pp. 296-300.

Shirai, S., Yamamoto, K. 2001. Linking English words in two bilingual dictionaries to generate another pair dictionary, *ICCPOL-2001*, pp. 174-179.

Stamou, S., Oflazer, K., Pala, K., Christoudoulakis, D., Cristea, D., Tufiş, D., Koeva, S., Totkov, G., Dutoit, D., Grigoriadou, M. 1997. BalkaNet: A Multilingual Semantic Network for the Balkan Languages, *In Proceedings of the International Wordnet Conference*, Mysore, India.

Tanaka, K., Umemura, K. 1994. Construction of a bilingual dictionary intermediated by a third language, *Proceedings of COLING-94*, pp. 297-303.

Vossen, P. 1998. Introduction to EuroWordNet. *Computers and the Humanities* 32: 73-89 Special Issue on EuroWordNet.

Multilingual Spectral Clustering Using Document Similarity Propagation

Dani Yogatama and **Kumiko Tanaka-Ishii**
Graduate School of Information Science and Technology, University of Tokyo
13F Akihabara Daibiru, 1-18-13 Kanda Chiyoda-ku, Tokyo, Japan
yogatama@cl.ci.i.u-tokyo.ac.jp kumiko@i.u-tokyo.ac.jp

Abstract

We present a novel approach for multilingual document clustering using only comparable corpora to achieve cross-lingual semantic interoperability. The method models document collections as weighted graph, and supervisory information is given as sets of must-linked constraints for documents in different languages. Recursive k-nearest neighbor similarity propagation is used to exploit the prior knowledge and merge two language spaces. Spectral method is applied to find the best cuts of the graph. Experimental results show that using limited supervisory information, our method achieves promising clustering results. Furthermore, since the method does not need any language dependent information in the process, our algorithm can be applied to languages in various alphabetical systems.

1 Introduction

Document clustering is unsupervised classification of text collections into distinct groups of similar documents. It has been used in many information retrieval tasks, including data organization (Siersdorfer and Sizov, 2004), language modeling (Liu and Croft, 2004), and improving performances of text categorization system (Aggarwal et al., 1999). Advance in internet technology has made the task of managing multilingual documents an intriguing research area. The growth of internet leads to the necessity of organizing documents in various languages. There exist thousands of languages, not to mention countless minor ones. Creating document clustering model for each language is simply unfeasible. We need methods to deal with text collections in diverse languages simultaneously.

Multilingual document clustering (MLDC) involves partitioning documents, written in more than one languages, into sets of clusters. Similar documents, even if they are written in different languages, should be grouped together into one cluster. The major challenge of MLDC is achieving cross-lingual semantic interoperability. Most monolingual techniques will not work since documents in different languages are mapped into different spaces. Spectral method such as Latent Semantic Analysis has been commonly applied for MLDC task. However, current techniques strongly rely on the presence of common words between different languages. This method would only work if the languages are highly related, i.e., languages that share the same root. Therefore, we need another method to improve the robustness of MLDC model.

In this paper, we focus on the problem of bridging multilingual space for document clustering. We are given text documents in different languages and asked to group them into clusters such that documents that belong to the same topic are grouped together. Traditional monolingual approach is impracticable since it is unable to predict how similar two multilingual documents are. They have two different spaces which make conventional cosine similarity irrelevant. We try to solve this problem utilizing prior knowledge in the form of must-linked constraints, gathered from comparable corpora. Propagation method is used to guide the language-space merging process. Experimental results show that the approach gives encouraging clustering results.

This paper is organized as follows. In section 2, we review related work. In section 3, we propose our algorithm for multilingual document clustering. The experimental results are shown in section 4. Section 5 concludes with a summary.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 871–879,
Singapore, 6-7 August 2009.

2 Related Work

Chen and Lin (2000) proposed methods to cluster multilingual documents using translation technology, relying on cross-lingual dictionary and machine-translation system. Multilingual ontology, such as Eurovoc, is also popular for MLDC (Pouliquen et al., 2004). However, such resources are scarce and expensive to build. Several other drawbacks of using this technique include dictionary limitation and word ambiguity.

More recently, parallel texts have been used to connect document collections from different languages (Wei et al., 2008). This is done by collapsing columns in a term by document matrix that are translations of each other. Nevertheless, building parallel texts is also expensive and requires a lot of works, hence shifting the paradigm of multilingual works to comparable corpora.

Comparable corpora are collections of texts in different languages regarding similar topics produced at the same time. The key difference between comparable corpora and parallel texts is that documents in comparable corpora are not necessarily translations of each other. They are easier to be acquired, and do not need exhaustive works to be prepared. News agencies often give information in many different languages and can be good sources for comparable corpora. Terms in comparable corpora, being about the same topic, up to some point explain the same concepts in different languages. Pairing comparable corpora with spectral method such as Latent Semantic Analysis has become prevalent, e.g. (Gliozzo and Strapparava, 2005). They rely on the presence of common words and proper nouns among various languages to build a language-independent space. The performance of such method is highly dependent on the languages being used. Here, we present another approach to exploit knowledge in comparable corpora; using propagation method to aid spreading similarity between collections of documents in different languages.

Spectral clustering is the task of finding good clusters by using information contained in the eigenvectors of a matrix derived from the data. It has been successfully applied in many applications including information retrieval (Deerwester et al., 2003) and computer vision (Meila and Shi, 2000). An in-depth analysis of spectral algorithm for clustering problems is given in (Ng et al., 2002). Zhang and Mao (2008) used a related technique called Modularity Eigenmap to extract community structure features from the document network to solve hypertext classification problem.

Semi-supervised clustering enhances clustering task by incorporating prior knowledge to aid clustering process. It allows user to guide the clustering process by giving some feedback to the model. In traditional clustering algorithm, only unlabeled data is used to find assignments of data points to clusters. In semi-supervised clustering, prior knowledge is given to improve performance of the system. The supervision is usually given as pair of must-linked constraints and cannot link constraints, first introduced in (Wagstaff and Cardie, 2000). Kamvar et al. (2003) proposed spectral learning algorithm that can take supervisory information in the form of pairwise constraints or labeled data. Their algorithm is intended to be used in monolingual context, while our algorithm is designed to work in multilingual context.

3 Multilingual Spectral Clustering

There have been several works on multilingual document clustering as mention previously in Section 2. Our key contribution here is the propagation method to make spectral clustering algorithm works for multilingual problems. The clustering model exploits the supervisory information by detecting k nearest neighbors of the newly-linked documents, and propagates document similarity to these neighbors. The model can be applied to any multilingual text collections regardless of the languages. Overall algorithm is given in Section 3.1 and the method to merge multilingual spaces by similarity propagation is given in Section 3.2.

3.1 Spectral Clustering Algorithm

Spectral clustering tries to find good clusters by using top eigenvectors of normalized data affinity matrix. The document set is being modeled as undirected graph $G(V, E, W)$, where V, E, and W denote the graph vertex set, edge set, and transition probability matrix, respectively. In graph G, $v \in V$ represents a document, and weight $w_{ij} \in W$ represents transition probability between document v_i to v_j. The transition probabilities can be interpreted as edge flows in Markov random walk over graph vertices (documents in collections).

Algorithm to perform spectral clustering is given in Algorithm 1. Let A be affinity matrix

where element A_{ij} is cosine similarity between document v_i and v_j (Algorithm 1, line 1). It is straightforward that documents belonging to different languages will have similarity zero. Rare exception occurs when they have common words because the languages are related one another. As a consequence, the similarity matrix will have many zeros. Our model amplifies prior knowledge in the form of comparable corpora by performing document similarity propagation, presented in Section 3.2 (Algorithm 1, line 4; Algorithm 2, explained in Section 3.2). After propagation, the affinity matrix is post-processed (Algorithm 1, line 6, explained in Section 3.2) before being transformed into transition probability matrix.

The transformation can be done using any normalization for spectral methods. Define $N = D^{-1}A$, as in (Meila and Shi, 2001), where D is the diagonal matrix whose elements $D_{ij} = \sum_j A_{ij}$ (Algorithm 1, line 7). Alternatively, we can define $N = D^{-1/2}AD^{-1/2}$ (Ng et al., 2002), or $N = (A + d_{max}I - D)/d_{max}$ (Fiedler, 1975), where d_{max} is the maximum rowsum of A. For our experiment, we use the first normalization method, though other methods can be applied as well.

Meila and Shi (2001) show that probability transition matrix N with t strong clusters will have t piecewise constant eigenvectors. They also suggest using these t eigenvectors in clustering process. We use the information contains in t largest eigenvectors of N (Algorithm 1, line 8-11) and perform K-means clustering algorithm to find the data clusters (Algorithm 1, line 12).

Algorithm 1 Multilingual Spectral Clustering

Input: Term by document matrix M, pairwise constraints

Output: Document clusters

1: Create graph affinity matrix $A \in \mathbb{R}^{n \times n}$ where each element A_{ij} represents the similarity between document v_i and v_j.
2: **for all** pairwise constraints in comparable corpora **do**
3: $A_{ij} \leftarrow 1, A_{ji} \leftarrow 1$.
4: Recursive Propagation $(A, S, \beta, k, v_i, v_j)$.
5: **end for**
6: Post-process matrix A so that every value in A is greater than δ and less than 1.
7: Form a diagonal matrix D, where $D_{ii} = \sum_j A_{ij}$. Normalize $N = D^{-1}A$.
8: Find $x_1, x_2 \cdots, x_t$, the t largest eigenvectors of N.
9: Form matrix $X = [x_1, x_2, \cdots, x_t] \in \mathbb{R}^{n \times t}$.
10: Normalize row X to be unit length.
11: Project each document into eigen-space spanned by the above t eigenvectors (by treating each row of X as a point in $\mathbb{R}^t$, row i represents document v_i).
12: Apply K-means algorithm in this space to find document clusters.

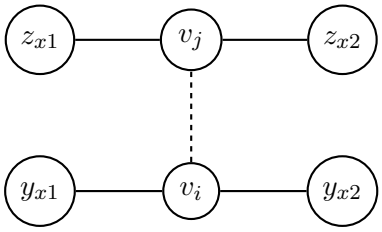
3.2 Propagating Prior Knowledge

We use information obtained from comparable corpora to merge multilingual language spaces. Suppose we have text collections in L different languages. We combine this collections with comparable corpora, also in L languages, that act as our supervisory information. Comparable corpora are used to gather prior knowledge by making must-linked constraints for documents in different languages that belong to the same topic in the corpora, propagating similarity to other documents while doing so.

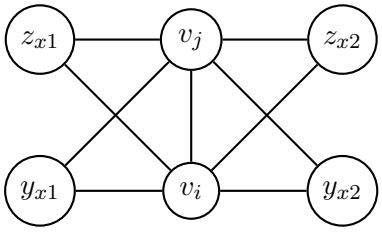
Initially, our affinity matrix A represents cosine similarity between all pairs of documents. A_{ij} is set to zero if j is not the top k nearest neighbors of i and likewise. Next, set A_{ij} and A_{ji} to 1 if document i and document j are different in language and belong to the same topic in our comparable corpora. This will incorporate the must-linked constraint to our model. We can also give supervisory information for pairs of document in the same language, but this is optional. We also do not use cannot-linked constraints since the main goal is to merge multilingual spaces. In our experiment we show that using only must-linked constraints with propagation is enough to achieve encouraging clustering results.

The supervisory information acquired from comparable corpora only connects two nodes in our graph. Therefore, the number of edges between documents in different languages is about as many as the number of must-linked constraints given. We argue that we need more edges between pairs of documents in different languages to get better results.

We try to build more edges by propagating similarity to other documents that are most similar to the newly-linked documents. Figure 1 gives an illustration of edge-creation process when two multilingual documents (nodes) are connected. Sup-



(a) Connect two nodes



(b) Effect on neighbor nodes

Figure 1: Pairing two multilingual documents affect their neighbors. v_i and v_j are documents in two different languages. y_x and z_x are neighbors of v_i and v_j respectively.

pose that we have six documents in two different languages. Initially, documents are only connected with other documents that belong to the same language. The supervisory information tells us that two multilingual documents v_i and v_j should be connected (Figure 1(a)). We then build an edge between these two documents. Furthermore, we also use this information to build edges between v_i and neighbors of v_j and likewise (Figure 1(b)).

This follows from the hypothesis that bringing together two documents should also bring other documents that are similar to those two closer in our clustering space. Klein et al. (2002) stated that a good clustering algorithm, besides satisfying known constraints, should also be able to satisfy the implications of those constraints. Here, we allow not only instance-level inductive implications, but utilize it to get higher-level inductive implications. In other words, we alter similarity space so that it can detect other clusters by changing the topology of the original space.

The process is analogous to shortening the distance between sets of documents in Euclidean space. In vector space model, two documents that are close to each other have high similarity, and thus will belong to the same cluster. Pairing two documents can be seen as setting the distance in this space to 0, thus raising their similarity to 1. While doing so, each document would also draw sets of documents connected to it closer to the centre of the merge, which is equivalent to increasing their similarities.

Suppose we have document v_i and v_j, and y and z are sets of their respective k nearest neighbors, where $|y| = |z| = k$. The propagation method is a recursive algorithm with base S, the number of desired level of propagation. Recursive k-nearest neighbor makes decision to give high similarity between multilingual documents not only determined by their similarity to the newly-linked documents, but also their similarity to the k nearest neighbors of the respective document. Several documents are affected by a single supervisory information. This will prove useful when only limited amount of supervisory information given. It uses document similarity matrix A, as defined in the previous section.

1. For $y_x \in y$ we propagate $\beta A_{v_i y_x}$ to $A_{v_j y_x}$. Set $A_{y_x v_j} = A_{v_j y_x}$ (Algorithm 2, line 5-6). In other words, we propagate the similarity between document v_i and y nearest neighbors of v_i to document v_j.

2. Similarly, for $z_x \in z$ we propagate $\beta A_{v_j z_x}$ to $A_{v_i z_x}$. Set $A_{z_x v_i} = A_{v_i z_x}$ (Algorithm 2, line 10-11). In other words, we propagate the similarity between document v_j and z nearest neighbors of v_j to document v_i.

3. Propagate higher order similarity to k nearest neighbors of y and z, discounting the similarity quadratically, until required level of propagation S is reached (Algorithm 2, line 7 and 12).

The coefficient β represents the degree of enforcement that the documents similar to a document in one language, will also have high similarity with other document in other language that is paired up with its ancestor. On the other hand, k represents the number of documents that are affected by pairing up two multilingual documents. After propagation, similarity of documents that falls below some threshold δ is set to zero (Algorithm 1, line 6). This post-processing step is performed to nullify insignificant similarity values propagated to a document. Additionally, if there exists similarity of documents that is higher than one, it is set to one.

Algorithm 2 Recursive Propagation

Input: Affinity matrix A, level of propagation S, β, number of nearest neighbors k, document v_i and v_j

Output: Propagated affinity matrix

if $S = 0$ **then**
  return
else
  for all $y_x \in k$-NN document v_i **do**
    $A_{v_j y_x} \leftarrow A_{v_j y_x} + \beta A_{v_i y_x}$
    $A_{y_x v_j} \leftarrow A_{v_j y_x}$
    Recursive Propagation $(A, S-1, \beta^2, k, y_x, v_j)$
  end for
  for all $z_x \in k$-NN document v_j **do**
    Set $A_{v_i z_x} \leftarrow A_{v_i z_x} + \beta A_{v_j z_x}$
    Set $A_{z_x v_i} \leftarrow A_{v_i z_x}$
    Recursive Propagation $(A, S-1, \beta^2, k, v_i, z_x)$
  end for
end if

4 Performance Evaluation

The goals of empirical evaluation include (1) testing whether the propagation method can merge multilingual space and produce acceptable clustering results; (2) comparing the performance to spectral clustering method without propagation.

4.1 Data Description

We tested our model using Reuters Corpus Volume 2 (RCV2), a multilingual corpus containing news in thirteen different languages. For our experiment, three different languages: English, French, and Spanish; in six different topics: science, sports, disasters accidents, religion, health, and economy are used. We discarded documents with multiple category labels.

We do not apply any language specific preprocessing method to the raw text data. Monolingual TFIDF is used for feature weighting. All document vectors are then converted into unit vector by dividing by its length. Table 1 shows the average length of documents in our corpus.

4.2 Evaluation Metric

For our experiment, we used Rand Index (RI) which is a common evaluation technique for clustering task where the true class of unlabeled data is known. Rand Index measures the percentage of decisions that are correct, or simply the accuracy of the model. Rand Index is defined as:

$$RI = \frac{TP + TN}{TP + FP + TN + FN}$$

Rand Index penalizes false positive and false negative decisions during clustering. It takes into account decision that assign two similar documents to one cluster (TP), two dissimilar documents to different clusters (TN), two similar documents to different clusters (FN), and two dissimilar documents to one cluster (FP). We do not include links created by supervisory information when calculating true positive decisions and only consider the number of free decisions made.

We also used F_α-measure, the weighted harmonic mean of precision (P) and recall (R). F_α-measure is defined as:

$$F_\alpha = \frac{(\alpha^2 + 1)PR}{\alpha^2 P + R}$$
$$P = \frac{TP}{TP + FP}$$
$$R = \frac{TP}{TP + FN}$$

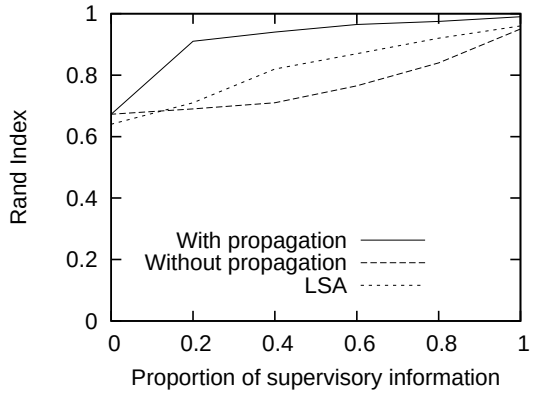
Last, we used purity to evaluate the accuracy of assignments. Purity is defined as:

$$Purity = \frac{1}{N} \sum_t \max_j |\omega_t \cap c_j|$$

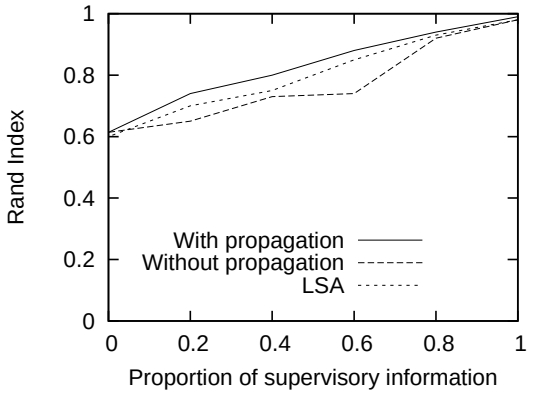
where N is the number of documents, t is the number of clusters, j is the number of classes, ω_t and c_j are sets of documents in cluster t and class j respectively.

	English	French	Spanish	Total
Science	290.10	165.10	213.45	222.88
Sports	182.55	156.83	189.75	176.37
Disasters	154.29	175.89	165.31	165.16
Religion	317.77	177.91	242.67	246.11
Health	251.19	233.70	227.25	237.38
Economy	266.89	192.55	306.11	255.08
Total	243.79	183.61	224.09	217.16

Table 1: Average number of words of documents in the corpus. Each language consists of 600 documents, and each topic consists of 100 documents (per language).

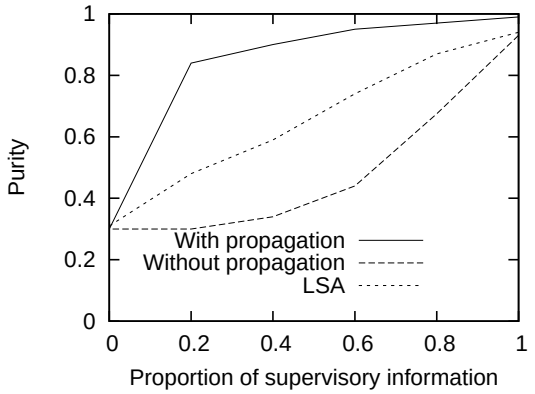


(a) Rand Index for 6 topics

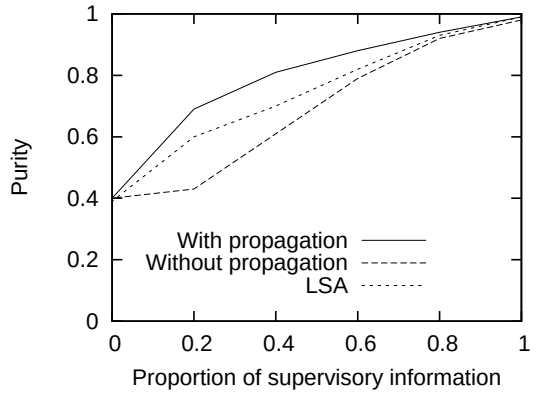


(b) Rand Index for 4 topics

Figure 2: Rand Index on the RCV2 task with (a) 1800 documents, 6 topics; and (b) 1200 documents, 4 topics as the proportion of supervisory information increases. $k = 30, \delta = 0.03, \beta = 0.5, t =$ number of topics, and $S = 2$.



(a) Purity for 6 topics

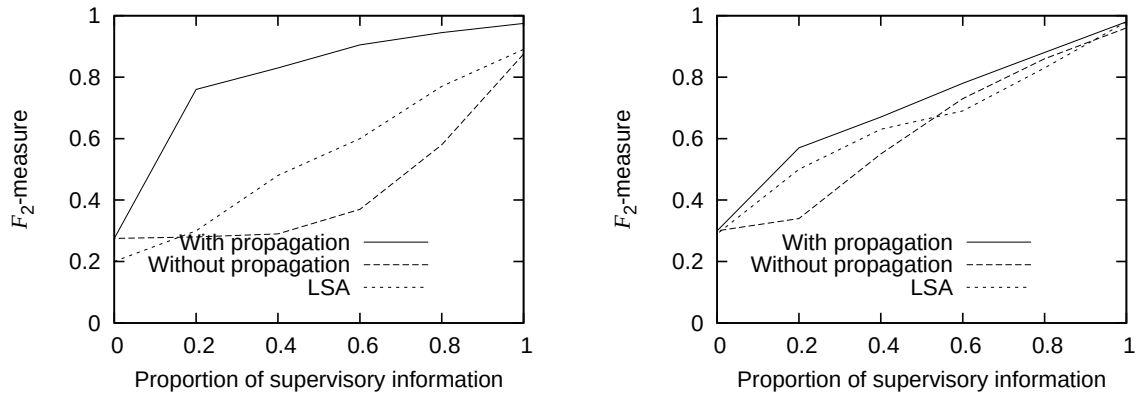


(b) Purity for 4 topics

Figure 3: Purity on the RCV2 task with (a) 1800 documents, 6 topics; and (b) 1200 documents, 4 topics as the proportion of supervisory information increases. $k = 30, \delta = 0.03, \beta = 0.5, t =$ number of topics, and $S = 2$.

4.3 Experimental Results

To prove the effectiveness of our clustering algorithm, we performed the following experiments on our data set. We first tested our algorithm on four topics, science, sports, religion, and economy. We then tested our algorithm using all six topics to get an understanding of the performance of our model in larger collections with more topics. We used subset of our data as supervisory information and built must-linked constraints from it. The proportion of supervisory information provided to the system is given in x-axis (Figure 2 - Figure 4.3). 0.2 here means 20% of documents in each language are taken to be used as prior knowledge. Since the number of documents in each language for our experiment is the same, we have the same numbers of documents in subset of English collection, subset of French collection, and subset of Spanish collection. We also ensure there are same numbers of documents for a particular topic in all three languages. We can build must-linked constraints as follows. For each document in the subset of English collection, we create must-linked constraints with one randomly selected document from the subset of French collection and one randomly selected document from the subset of Spanish collection that belong to the same topic with it. We then create must-linked constraint between the respective French and Spanish documents. The constraints given to the algorithm are chosen so that there are several links that connect every topic in every language. Note that the class label in-



(a) F_2-measure for 6 topics

(b) F_2-measure for 4 topics

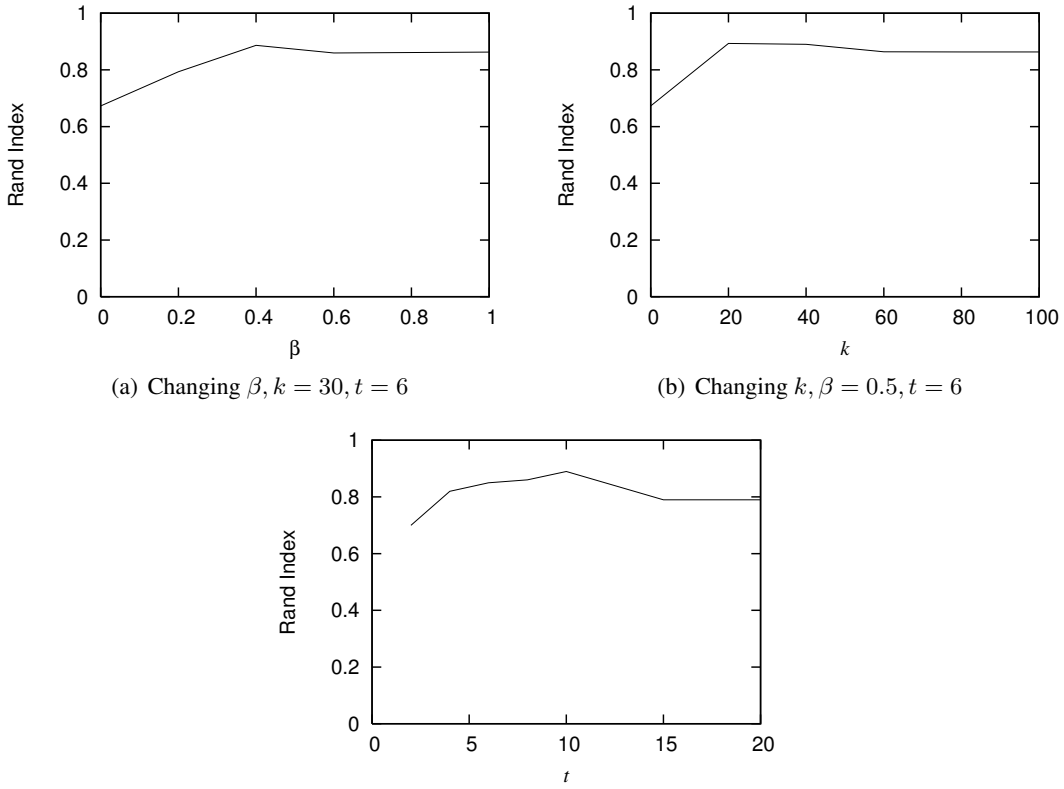
Figure 4: F_2-measure on the RCV2 task with (a) 1800 documents, 6 topics; and (b) 1200 documents, 4 topics as the proportion of supervisory information increases. $k = 30, \delta = 0.03, \beta = 0.5, t =$ number of topics, and $S = 2$.

formation is only used to build must-linked constraints between documents, and we do not assign the documents to a particular cluster.

Figure 2 shows the Rand Index as proportion of supervisory information increases. Figure 3 and Figure 4.3 give purity and F_2-measure for the algorithm respectively. To show the importance of the propagation in multilingual space, we give comparison with spectral clustering model without propagation. Three lines in Figure 2 to Figure 4.3 indicate: (1) results with propagation (solid line); (2) results without propagation (long-dashed line); and (3) results using Latent Semantic Analysis(LSA)-based method by exploiting common words between languages (short-dashed line). For each figure, 6 plots are taken starting from 0 in 0.2-point-increments. We conducted the experiments three times for each proportion of supervisory information and use the average values. As we can see from Figure 2, Figure 3, and Figure 4.3, the propagation method can significantly improve the performance of spectral clustering algorithm. For 1800 documents in 6 topics, we manage to achieve RI $= 0.91$, purity $= 0.84$, and F_2-measure $= 0.76$ with only 20% of documents (360 documents) used as supervisory information. Spectral clustering algorithm without propagation can only achieve $0.69, 0.30, 0.28$ for RI, purity, and F_2-measure respectively. The propagation method is highly effective when only small amount of supervisory information given to the algorithm. Obviously, the more supervisory information given, the better the performance is. As the number of supervisory information increases, the difference of the model performance with and without propagation becomes smaller. This is because there are already enough links between multilingual documents, so we do not necessarily build more links through similarity propagation anymore. However, even when there are already many links, our model with propagation still outperforms the model without propagation.

We compare the performance of our algorithm to LSA-based multilingual document clustering model. We performed LSA to the multilingual term by document matrix. We do not use parallel texts and only rely on common words across languages as well as must-linked constraints to build multilingual space. The results show that exploiting common words between languages alone is not enough to build a good multilingual semantic space, justifying the usage of supervisory information in multilingual document clustering task. When supervisory information is introduced, our method achieves better results than LSA-based method. In general, the LSA-based method performs better than the model without propagation.

We assess the sensitivity of our algorithm to parameter β, the penalty for similarity propagation. We assess the sensitivity of our algorithm to parameter β, the penalty for similarity propagation. We tested our algorithm using various β, starting from 0 to 1 in 0.2-point-increments, while other parameters being held constant. Figure 5(a) shows that changing β to some extent affects the performance of the algorithm. However, after some value of reasonable β is found, increasing β does not have significant impact on the per-



(a) Changing $\beta, k = 30, t = 6$

(b) Changing $k, \beta = 0.5, t = 6$

(c) Changing $t, \beta = 0.5, k = 30$

Figure 5: Rand Index on the RCV2 task with 1800 documents and 6 topics as (a) β increases; (b) k increases; and (c) t increases. $\delta = 0.03, S = 2$, and 20% of documents are used as supervisory information.

formance of the algorithm. We also tested our algorithm using various k, starting from 0 to 100 in 20-point-increments. Figure 5(b) reveals that the performances of the model with different k are comparable, as long as k is not too small. However, using too large k will slightly decrease the performance of the model. Too many propagations make several dissimilar documents receive high similarity value that cannot be nullified by the post-processing step. Last, we experimented using various t ranging from 2 to 20. Figure 5(c) shows that the method performs best when $t = 10$, and for reasonable value of t the method achieves comparable performance.

5 Conclusion

We present here a multilingual spectral clustering model that is able to work irrespective of the languages being used. The key component of our model is the propagation algorithm to merge multilingual spaces. We tested our algorithm on Reuters RCV2 Corpus and compared the performance with spectral clustering model without propagation. Experimental results reveal that using limited supervisory information, the algorithm achieves encouraging clustering results.

References

Charu C. Aggarwal, Stephen C. Gates and Philip S. Yu. 1999. On The Merits of Building Categorization Systems by Supervised Clustering. In *Proceedings of Conference on Knowledge Discovery in Databases*:352-356.

Hsin-Hsi Chen and Chuan-Jie Lin. 2000. A Multilingual News Summarizer. In *Proceedings of 18th International Conference on Computational Linguistics*:159-165.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*:41(6):391-407.

Miroslav Fiedler. 1975. A Property of Eigenvectors of Nonnegative Symmetric Matrices and its Applications to Graph Theory. *Czechoslovak Mathematical Journal*, 25:619-672.

Alfio Gliozzo and Carlo Strapparava. 2005. Cross language Text Categorization by acquiring Multilingual Domain Models from Comparable Corpora. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*:9-16.

Sepandar D. Kamvar, Dan Klein, and Christopher D. Manning. 2003. Spectral Learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. 2002. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *The Nineteenth International Conference on Machine Learning*.

Xiaoyong Liu and W. Bruce Croft. 2004. Cluster-based Retrieval using Language Models. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*:186-193.

Marinla Meilă and Jianbo Shi. 2000. Learning segmentation by random walks. In *Advances in Neural Information Processing Systems*:873-879.

Marinla Meilă and Jianbo Shi. 2001. A Random Walks View of Spectral Segmentation. In *AI and Statistics (AISTATS)*.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2002. On Spectral Clustering: Analysis and an algorithm. In *Proceedings of Advances in Neural Information Processing Systems (NIPS 14)*.

Bruno Pouliquen, Ralf Steinberger, Camelia Ignat, Emilia Käsper, and Irina Temnikova. 2004. Multilingual and Cross-lingual News Topic Tracking. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Stefan Siersdorfer and Sergej Sizov. 2004. Restrictive Clustering and Metaclustering for Self-Organizing Document. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*.

Kiri Wagstaff and Claire Cardie 2000. Clustering with Instance-level Constraints. In *Proceedings of the 17th International Conference on Machine Learning*:1103-1110.

Chih-Ping Wei, Christopher C. Yang, and Chia-Min Lin. 2008. A Latent Semantic Indexing Based Approach to Multilingual Document Clustering. In *Decision Support Systems*, 45(3):606-620

Dell Zhang and Robert Mao. 2008. Extracting Community Structure Features for Hypertext Classification. In *Proceedings of the 3rd IEEE International Conference on Digital Information Management (ICDIM)*.

Polylingual Topic Models

David Mimno Hanna M. Wallach Jason Naradowsky David A. Smith Andrew McCallum
University of Massachusetts, Amherst
Amherst, MA 01003
{mimno, wallach, narad, dasmith, mccallum}cs.umass.edu

Abstract

Topic models are a useful tool for analyzing large text collections, but have previously been applied in only monolingual, or at most bilingual, contexts. Meanwhile, massive collections of interlinked documents in dozens of languages, such as Wikipedia, are now widely available, calling for tools that can characterize content in many languages. We introduce a polylingual topic model that discovers topics aligned across multiple languages. We explore the model's characteristics using two large corpora, each with over ten different languages, and demonstrate its usefulness in supporting machine translation and tracking topic trends across languages.

1 Introduction

Statistical topic models have emerged as an increasingly useful analysis tool for large text collections. Topic models have been used for analyzing topic trends in research literature (Mann et al., 2006; Hall et al., 2008), inferring captions for images (Blei and Jordan, 2003), social network analysis in email (McCallum et al., 2005), and expanding queries with topically related words in information retrieval (Wei and Croft, 2006). Much of this work, however, has occurred in monolingual contexts. In an increasingly connected world, the ability to access documents in many languages has become both a strategic asset and a personally enriching experience. In this paper, we present the polylingual topic model (PLTM). We demonstrate its utility and explore its characteristics using two polylingual corpora: proceedings of the European parliament (in eleven languages) and a collection of Wikipedia articles (in twelve languages).

There are many potential applications for polylingual topic models. Although research literature is typically written in English, bibliographic databases often contain substantial quantities of work in other languages. To perform topic-based bibliometric analysis on these collections, it is necessary to have topic models that are aligned across languages. Such analysis could be significant in tracking international research trends, where language barriers slow the transfer of ideas.

Previous work on bilingual topic modeling has focused on machine translation applications, which rely on sentence-aligned parallel translations. However, the growth of the internet, and in particular Wikipedia, has made vast corpora of *topically* comparable texts—documents that are topically similar but are not direct translations of one another—considerably more abundant than ever before. We argue that topic modeling is both a useful and appropriate tool for leveraging correspondences between semantically comparable documents in multiple different languages.

In this paper, we use two polylingual corpora to answer various critical questions related to polylingual topic models. We employ a set of direct translations, the EuroParl corpus, to evaluate whether PLTM can accurately infer topics when documents genuinely contain the same content. We also explore how the characteristics of different languages affect topic model performance. The second corpus, Wikipedia articles in twelve languages, contains sets of documents that are not translations of one another, but are very likely to be about similar concepts. We use this corpus to explore the ability of the model both to infer similarities between vocabularies in different languages, and to detect differences in topic emphasis between languages. The internet makes it possible for people all over the world to access documents from different cultures, but readers will not be fluent in this wide variety of languages. By linking topics across languages, polylingual topic models can increase cross-cultural understanding by providing readers with the ability to characterize

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 880–889,
Singapore, 6-7 August 2009.

the contents of collections in unfamiliar languages and identify trends in topic prevalence.

2 Related Work

Bilingual topic models for parallel texts with word-to-word alignments have been studied previously using the HM-bitam model (Zhao and Xing, 2007). Tam, Lane and Schultz (Tam et al., 2007) also show improvements in machine translation using bilingual topic models. Both of these translation-focused topic models infer word-to-word alignments as part of their inference procedures, which would become exponentially more complex if additional languages were added. We take a simpler approach that is more suitable for topically similar document tuples (where documents are not direct translations of one another) in more than two languages. A recent extended abstract, developed concurrently by Ni et al. (Ni et al., 2009), discusses a multilingual topic model similar to the one presented here. However, they evaluate their model on only two languages (English and Chinese), and do not use the model to detect differences between languages. They also provide little analysis of the differences between polylingual and single-language topic models. Outside of the field of topic modeling, Kawaba et al. (Kawaba et al., 2008) use a Wikipedia-based model to perform sentiment analysis of blog posts. They find, for example, that English blog posts about the Nintendo Wii often relate to a hack, which cannot be mentioned in Japanese posts due to Japanese intellectual property law. Similarly, posts about whaling often use (positive) nationalist language in Japanese and (negative) environmentalist language in English.

3 Polylingual Topic Model

The polylingual topic model (PLTM) is an extension of latent Dirichlet allocation (LDA) (Blei et al., 2003) for modeling polylingual document tuples. Each tuple is a set of documents that are loosely equivalent to each other, but written in different languages, e.g., corresponding Wikipedia articles in French, English and German. PLTM assumes that the documents in a tuple share the same tuple-specific distribution over topics. This is unlike LDA, in which each document is assumed to have its own document-specific distribution over topics. Additionally, PLTM assumes that each "topic" consists of a *set* of discrete distributions

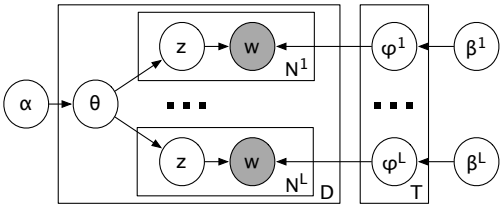


Figure 1: Graphical model for PLTM.

over words—one for each language $l = 1, \ldots, L$. In other words, rather than using a single set of topics $\Phi = \{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_T\}$, as in LDA, there are L sets of language-specific topics, $\Phi^1, \ldots, \Phi^L$, each of which is drawn from a language-specific symmetric Dirichlet with concentration parameter β^l.

3.1 Generative Process

A new document tuple $\boldsymbol{w} = (\boldsymbol{w}^1, \ldots, \boldsymbol{w}^L)$ is generated by first drawing a tuple-specific topic distribution from an asymmetric Dirichlet prior with concentration parameter α and base measure $\boldsymbol{m}$:

$$\boldsymbol{\theta} \sim \text{Dir}\,(\boldsymbol{\theta}, \alpha \boldsymbol{m}). \tag{1}$$

Then, for each language l, a latent topic assignment is drawn for each token in that language:

$$\boldsymbol{z}^l \sim P(\boldsymbol{z}^l \,|\, \boldsymbol{\theta}) = \textstyle\prod_n \theta_{z_n^l}. \tag{2}$$

Finally, the observed tokens are themselves drawn using the language-specific topic parameters:

$$\boldsymbol{w}^l \sim P(\boldsymbol{w}^l \,|\, \boldsymbol{z}^l, \Phi^l) = \textstyle\prod_n \phi^l_{w_n^l | z_n^l}. \tag{3}$$

The graphical model is shown in figure 1.

3.2 Inference

Given a corpus of training and test document tuples—$\mathcal{W}$ and $\mathcal{W}'$, respectively—two possible inference tasks of interest are: computing the probability of the test tuples given the training tuples and inferring latent topic assignments for test documents. These tasks can either be accomplished by averaging over samples of $\Phi^1, \ldots, \Phi^L$ and $\alpha\boldsymbol{m}$ from $P(\Phi^1, \ldots, \Phi^L, \alpha\boldsymbol{m} \,|\, \mathcal{W}', \beta)$ or by evaluating a point estimate. We take the latter approach, and use the MAP estimate for $\alpha\boldsymbol{m}$ and the predictive distributions over words for $\Phi^1, \ldots, \Phi^L$. The probability of held-out document tuples $\mathcal{W}'$ given training tuples $\mathcal{W}$ is then approximated by $P(\mathcal{W}' \,|\, \Phi^1, \ldots, \Phi^L, \alpha\boldsymbol{m})$.

Topic assignments for a test document tuple $\boldsymbol{w} = (\boldsymbol{w}^1, \ldots, \boldsymbol{w}^L)$ can be inferred using Gibbs

sampling. Gibbs sampling involves sequentially resampling each z_n^l from its conditional posterior:

$$P(z_n^l = t \mid \boldsymbol{w}, \boldsymbol{z}_{\setminus l,n}, \Phi^1, \ldots, \Phi^L, \alpha\boldsymbol{m}) \propto \phi^l_{w_n^l|t} \frac{(N_t)_{\setminus l,n} + \alpha m_t}{\sum_t N_t - 1 + \alpha}, \quad (4)$$

where $\boldsymbol{z}_{\setminus l,n}$ is the current set of topic assignments for all other tokens in the tuple, while $(N_t)_{\setminus l,n}$ is the number of occurrences of topic t in the tuple, excluding z_n^l, the variable being resampled.

4 Results on Parallel Text

Our first set of experiments focuses on document tuples that are known to consist of direct translations. In this case, we can be confident that the topic distribution is genuinely shared across all languages. Although direct translations in multiple languages are relatively rare (in contrast with comparable documents), we use direct translations to explore the characteristics of the model.

4.1 Data Set

The EuroParl corpus consists of parallel texts in eleven western European languages: Danish, German, Greek, English, Spanish, Finnish, French, Italian, Dutch, Portuguese and Swedish. These texts consist of roughly a decade of proceedings of the European parliament. For our purposes we use alignments at the speech level rather than the sentence level, as in many translation tasks using this corpus. We also remove the twenty-five most frequent word types for efficiency reasons. The remaining collection consists of over 121 million words. Details by language are shown in Table 1.

Table 1: Average document length, # documents, and unique word types per 10,000 tokens in the EuroParl corpus.

Lang.	Avg. leng.	# docs	types/10k
DA	160.153	65245	121.4
DE	178.689	66497	124.5
EL	171.289	46317	124.2
EN	176.450	69522	43.1
ES	170.536	65929	59.5
FI	161.293	60822	336.2
FR	186.742	67430	54.8
IT	187.451	66035	69.5
NL	176.114	66952	80.8
PT	183.410	65718	68.2
SV	154.605	58011	136.1

Models are trained using 1000 iterations of Gibbs sampling. Each language-specific topic–word concentration parameter β^l is set to 0.01.

DA centralbank europæiske ecb s lån centralbanks
DE zentralbank ezb bank europäischen investitionsbank darlehen
EL τράπεζα τράπεζας κεντρική εκτ κεντρικής τράπεζες
EN **bank central ecb banks european monetary**
ES banco central europeo bce bancos centrales
FI keskuspankin ekp n euroopan keskuspankki eip
FR banque centrale bce européenne banques monétaire
IT banca centrale bce europea banche prestiti
NL bank centrale ecb europese banken leningen
PT banco central europeu bce bancos empréstimos
SV centralbanken europeiska ecb centralbankens s lån

DA børn familie udnyttelse børns børnene seksuel
DE kinder kindern familie ausbeutung familien eltern
EL παιδιά παιδιών οικογένεια οικογένειας γονείς παιδικής
EN **children family child sexual families exploitation**
ES niños familia hijos sexual infantil menores
FI lasten lapsia lapset perheen lapsen lapsiin
FR enfants famille enfant parents exploitation familles
IT bambini famiglia figli minori sessuale sfruttamento
NL kinderen kind gezin seksuele ouders familie
PT crianças família filhos sexual criança infantil
SV barn barnen familjen sexuellt familj utnyttjande

DA mål nå målsætninger målet målsætning opnå
DE ziel ziele erreichen zielen erreicht zielsetzungen
EL στόχους στόχο στόχος στόχων στόχοι επίτευξη
EN **objective objectives achieve aim ambitious set**
ES objetivo objetivos alcanzar conseguir lograr estos
FI tavoite tavoitteet tavoitteena tavoitteiden tavoitteita tavoitteen
FR objectif objectifs atteindre but cet ambitieux
IT obiettivo obiettivi raggiungere degli scopo quello
NL doelstellingen doel doelstelling bereiken bereikt doelen
PT objectivo objectivos alcançar atingir ambicioso conseguir
SV mål målet uppnå målen målsättningar målsättning

DA andre anden side ene andet øvrige
DE anderen andere einen wie andererseits anderer
EL άλλες άλλα άλλη άλλων άλλους όπως
EN **other one hand others another there**
ES otros otras otro otra parte demás
FI muiden toisaalta muita muut muihin muun
FR autres autre part côté ailleurs même
IT altri altre altro altra dall parte
NL andere anderzijds anderen ander als kant
PT outros outras outro lado outra noutros
SV andra sidan å annat ena annan

Figure 2: EuroParl topics (T=400)

The concentration parameter α for the prior over document-specific topic distributions is initialized to $0.01\,T$, while the base measure $\boldsymbol{m}$ is initialized to the uniform distribution. Hyperparameters $\alpha\boldsymbol{m}$ are re-estimated every 10 Gibbs iterations.

4.2 Analysis of Trained Models

Figure 2 shows the most probable words in all languages for four example topics, from PLTM with 400 topics. The first topic contains words relating to the European Central Bank. This topic provides an illustration of the variation in technical terminology captured by PLTM, including the wide array of acronyms used by different languages. The second topic, concerning children, demonstrates the variability of everyday terminology: although the four Romance languages are closely

related, they use etymologically unrelated words for children. (Interestingly, all languages except Greek and Finnish use closely related words for "youth" or "young" in a separate topic.) The third topic demonstrates differences in inflectional variation. English and the Romance languages use only singular and plural versions of "objective." The other Germanic languages include compound words, while Greek and Finnish are dominated by inflected variants of the same lexical item. The final topic demonstrates that PLTM effectively clusters "syntactic" words, as well as more semantically specific nouns, adjectives and verbs.

Although the topics in figure 2 seem highly focused, it is interesting to ask whether the model is genuinely learning mixtures of topics or simply assigning entire document tuples to single topics. To answer this question, we compute the posterior probability of each topic in each tuple under the trained model. If the model assigns all tokens in a tuple to a single topic, the maximum posterior topic probability for that tuple will be near to 1.0. If the model assigns topics uniformly, the maximum topic probability will be near $1/T$. We compute histograms of these maximum topic probabilities for $T \in \{50, 100, 200, 400, 800\}$. For clarity, rather than overlaying five histograms, figure 3 shows the histograms converted into smooth curves using a kernel density estimator.[1] Although there is a small bump around 1.0 (for extremely short documents, e.g., "Applause"), values are generally closer to, but greater than, $1/T$.

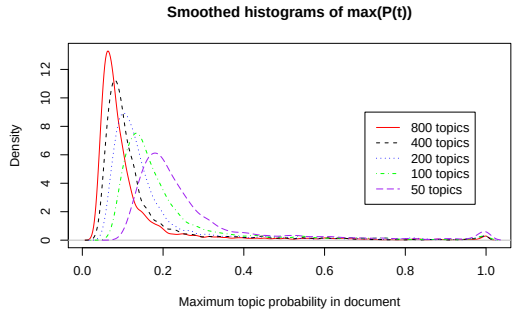


Figure 3: Smoothed histograms of the probability of the most probable topic in a document tuple.

Although the posterior distribution over topics for each tuple is not concentrated on one topic, it is worth checking that this is not simply because the model is assigning a single topic to the tokens in each of the languages. Although the model does not distinguish between topic assignment variables within a given document tuple (so it is technically incorrect to speak of different posterior distributions over topics for different documents in a given tuple), we can nevertheless divide topic assignment variables between languages and use them to estimate a Dirichlet-multinomial posterior distribution for each language in each tuple. For each tuple we can then calculate the Jensen-Shannon divergence (the average of the KL divergences between each distribution and a mean distribution) between these distributions. Figure 4 shows the density of these divergences for different numbers of topics. As with the previous figure, there are a small number of documents that contain only one topic in all languages, and thus have zero divergence. These tend to be very short, formulaic parliamentary responses, however. The vast majority of divergences are relatively low (1.0 indicates no overlap in topics between languages in a given document tuple) indicating that, for each tuple, the model is not simply assigning all tokens in a particular language to a single topic. As the number of topics increases, greater variability in topic distributions causes divergence to increase.

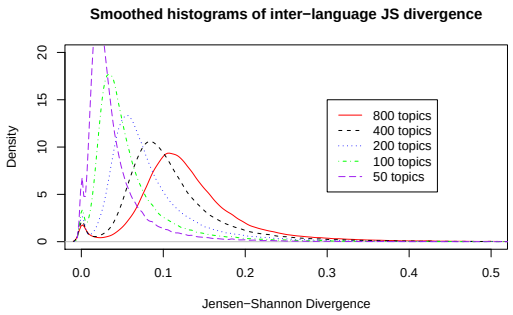


Figure 4: Smoothed histograms of the Jensen-Shannon divergences between the posterior probability of topics between languages.

4.3 Language Model Evaluation

A topic model specifies a probability distribution over documents, or in the case of PLTM, document tuples. Given a set of training document tuples, PLTM can be used to obtain posterior estimates of $\Phi^1, \ldots, \Phi^L$ and $\alpha\boldsymbol{m}$. The probability of previously unseen held-out document tuples given these estimates can then be computed. The higher the probability of the held-out document tuples, the better the generalization ability of the model.

[1] We use the R `density` function.

Analytically calculating the probability of a set of held-out document tuples given $\Phi^1, \ldots, \Phi^L$ and $\alpha\boldsymbol{m}$ is intractable, due to the summation over an exponential number of topic assignments for these held-out documents. However, recently developed methods provide efficient, accurate estimates of this probability. We use the "left-to-right" method of (Wallach et al., 2009). We perform five estimation runs for each document and then calculate standard errors using a bootstrap method.

Table 2 shows the log probability of held-out data in nats per word for PLTM and LDA, both trained with 200 topics. There is substantial variation between languages. Additionally, the predictive ability of PLTM is consistently slightly worse than that of (monolingual) LDA. It is important to note, however, that these results do not imply that LDA should be preferred over PLTM—that choice depends upon the needs of the modeler. Rather, these results are intended as a quantitative analysis of the difference between the two models.

Table 2: Held-out log probability in nats/word. (Smaller magnitude implies better language modeling performance.) PLTM does slightly worse than monolingual LDA models, but the variation between languages is much larger.

Lang	PLTM	sd	LDA	sd
DA	-8.11	0.00067	-8.02	0.00066
DE	-8.17	0.00057	-8.08	0.00072
EL	-8.44	0.00079	-8.36	0.00087
EN	-7.51	0.00064	-7.42	0.00069
ES	-7.98	0.00073	-7.87	0.00070
FI	-9.25	0.00089	-9.21	0.00065
FR	-8.26	0.00072	-8.19	0.00058
IT	-8.11	0.00071	-8.02	0.00058
NL	-7.84	0.00067	-7.75	0.00099
PT	-7.87	0.00085	-7.80	0.00060
SV	-8.25	0.00091	-8.16	0.00086

As the number of topics is increased, the word counts per topic become very sparse in monolingual LDA models, proportional to the size of the vocabulary. Figure 5 shows the proportion of all tokens in English and Finnish assigned to each topic under LDA and PLTM with 800 topics. More than 350 topics in the Finnish LDA model have zero tokens assigned to them, and almost all tokens are assigned to the largest 200 topics. English has a larger tail, with non-zero counts in all but 16 topics. In contrast, PLTM assigns a significant number of tokens to almost all 800 topics, in very similar proportions in both languages. PLTM topics therefore have a higher granularity – i.e., they are more specific. This result is important: informally, we have found that increasing the granularity of topics correlates strongly with user perceptions of the utility of a topic model.

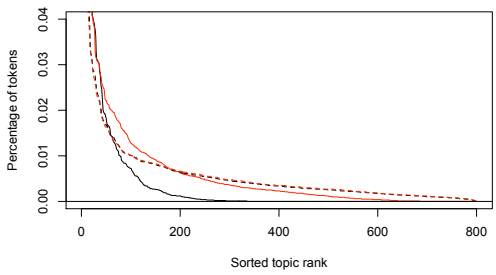


Figure 5: Topics sorted by number of words assigned. Finnish is in black, English is in red; LDA is solid, PLTM is dashed. LDA in Finnish essentially learns a 200 topic model when given 800 topics, while PLTM uses all 800 topics.

4.4 Partly Comparable Corpora

An important application for polylingual topic modeling is to use small numbers of comparable document tuples to link topics in larger collections of distinct, non-comparable documents in multiple languages. For example, a journal might publish papers in English, French, German and Italian. No paper is exactly comparable to any other paper, but they are all roughly topically similar. If we wish to perform topic-based bibliometric analysis, it is vital to be able to track the same topics across all languages. One simple way to achieve this topic alignment is to add a small set of comparable document tuples that provide sufficient "glue" to bind the topics together. Continuing with the example above, one might extract a set of connected Wikipedia articles related to the focus of the journal and then train PLTM on a joint corpus consisting of journal papers and Wikipedia articles.

In order to simulate this scenario we create a set of variations of the EuroParl corpus by treating some documents as if they have no parallel/comparable texts – i.e., we put each of these documents in a single-document tuple. To do this, we divide the corpus $\mathcal{W}$ into two sets of document tuples: a "glue" set $\mathcal{G}$ and a "separate" set $\mathcal{S}$ such that $|\mathcal{G}| \,/\, |\mathcal{W}| = p$. In other words, the proportion of tuples in the corpus that are treated as "glue" (i.e., placed in $\mathcal{G}$) is p. For every tuple in $\mathcal{S}$, we assign each document in that tuple to a new single-document tuple. By doing this, every document in $\mathcal{S}$ has its own distribution over topics, independent of any other documents. Ideally, the "glue" doc-

uments in $\mathcal{G}$ will be sufficient to align the topics across languages, and will cause comparable documents in $\mathcal{S}$ to have similar distributions over topics even though they are modeled independently.

Table 3: The effect of the proportion p of "glue" tuples on mean Jensen-Shannon divergence in estimated topic distributions for pairs of documents in $\mathcal{S}$ that were originally part of a document tuple. Lower divergence means the topic distributions distributions are more similar to each other.

p	Mean JS	# of pairs	Std. Err.
0.01	0.83755	487670	0.00018
0.05	0.79144	467288	0.00021
0.1	0.70228	443753	0.00026
0.25	0.38480	369608	0.00029
0.5	0.29712	246380	0.00030

Table 4: Topics are meaningful within languages but diverge between languages when only 1% of tuples are treated as "glue" tuples. With 25% "glue" tuples, topics are aligned.

lang	Topics at $p = 0.01$
DE	rußland russland russischen tschetschenien sicherheit
EN	china rights human country s burma
FR	russie tchétchénie union avec russe région
IT	ho presidente mi perché relazione votato

lang	Topics at $p = 0.25$
DE	rußland russland russischen tschetschenien ukraine
EN	russia russian chechnya cooperation region belarus
FR	russie tchétchénie avec russe russes situation
IT	russia unione cooperazione cecenia regione russa

We train PLTM with 100 topics on corpora with $p \in \{0.01, 0.05, 0.1, 0.25, 0.5\}$. We use 1000 iterations of Gibbs sampling with $\beta = 0.01$. Hyperparameters $\alpha\boldsymbol{m}$ are re-estimated every 10 iterations. We calculate the Jensen-Shannon divergence between the topic distributions for each pair of individual documents in $\mathcal{S}$ that were originally part of the same tuple prior to separation. The lower the divergence, the more similar the distributions are to each other. From the results in figure 4, we know that leaving all document tuples intact should result in a mean JS divergence of less than 0.1. Table 3 shows mean JS divergences for each value of p. As expected, JS divergence is greater than that obtained when all tuples are left intact. Divergence drops significantly when the proportion of "glue" tuples increases from 0.01 to 0.25. Example topics for $p = 0.01$ and $p = 0.25$ are shown in table 4. At $p = 0.01$ (1% "glue" documents), German and French both include words relating to Russia, while the English and Italian word distributions appear locally consistent but unrelated to Russia. At $p = 0.25$, the top words for all four languages are related to Russia.

These results demonstrate that PLTM is appropriate for aligning topics in corpora that have only a small subset of comparable documents. One area for future work is to explore whether initialization techniques or better representations of topic co-occurrence might result in alignment of topics with a smaller proportion of comparable texts.

4.5 Machine Translation

Although the PLTM is clearly not a substitute for a machine translation system—it has no way to represent syntax or even multi-word phrases—it is clear from the examples in figure 2 that the sets of high probability words in different languages for a given topic are likely to include translations. We therefore evaluate the ability of the PLTM to generate bilingual lexica, similar to other work in unsupervised translation modeling (Haghighi et al., 2008). In the early statistical translation model work at IBM, these representations were called "cepts," short for concepts (Brown et al., 1993).

We evaluate sets of high-probability words in each topic and multilingual "synsets" by comparing them to entries in human-constructed bilingual dictionaries, as done by Haghighi et al. (2008). Unlike previous work (Koehn and Knight, 2002), we evaluate all words, not just nouns. We collected bilingual lexica mapping English words to German, Greek, Spanish, French, Italian, Dutch and Swedish. Each lexicon is a set of pairs consisting of an English word and a translated word, $\{w_e, w_\ell\}$. We do not consider multi-word terms. We expect that simple analysis of topic assignments for sequential words would yield such collocations, but we leave this for future work.

For every topic t we select a small number K of the most probable words in English (e) and in each "translation" language (ℓ): $\mathcal{W}_{te}$ and $\mathcal{W}_{t\ell}$, respectively. We then add the Cartesian product of these sets for every topic to a set of candidate translations $\mathcal{C}$. We report the number of elements of $\mathcal{C}$ that appear in the reference lexica. Results for $K = 1$, that is, considering only the single most probable word for each language, are shown in figure 6. Precision at this level is relatively high, above 50% for Spanish, French and Italian with $T = 400$ and 800. Many of the candidate pairs that were not in the bilingual lexica were valid translations (e.g. EN "comitology" and IT

"comitalogia") that simply were not in the lexica. We also do not count morphological variants: the model finds EN "rules" and DE "vorschriften," but the lexicon contains only "rule" and "vorschrift." Results remain strong as we increase K. With $K = 3, T = 800$, 1349 of the 7200 candidate pairs for Spanish appeared in the lexicon.

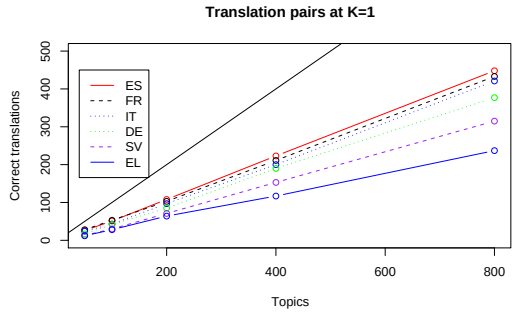


Figure 6: Are the single most probable words for a given topic in different languages translations of each other? The number of such pairs that appear in bilingual lexica is shown on the y-axis. For $T = 800$, the top English and Spanish words in 448 topics were exact translations of one another.

4.6 Finding Translations

In addition to enhancing lexicons by aligning topic-specific vocabulary, PLTM may also be useful for adapting machine translation systems to new domains by finding translations or near translations in an unstructured corpus. These aligned document pairs could then be fed into standard machine translation systems as training data. To evaluate this scenario, we train PLTM on a set of document tuples from EuroParl, infer topic distributions for a set of held-out documents, and then measure our ability to align documents in one language with their translations in another language.

It is not necessarily clear that PLTM will be effective at identifying translations. In finding a low-dimensional semantic representation, topic models deliberately smooth over much of the variation present in language. We are therefore interested in determining whether the information in the document-specific topic distributions is sufficient to identify semantically identical documents.

We begin by dividing the data into a training set of 69,550 document tuples and a test set of 17,435 document tuples. In order to make the task more difficult, we train a relatively coarse-grained PLTM with 50 topics on the training set. We then use this model to infer topic distributions for each

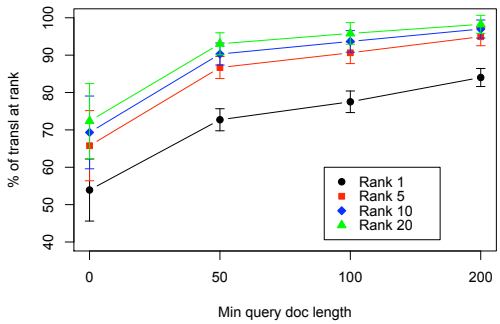


Figure 7: Percent of query language documents for which the target language translation is ranked at or above 1, 5, 10 or 20 by JS divergence, averaged over all language pairs.

of the 11 documents in each of the held-out document tuples using a method similar to that used to calculate held-out probabilities (Wallach et al., 2009). Finally, for each pair of languages ("query" and "target") we calculate the difference between the topic distribution for each held-out document in the query language and the topic distribution for each held-out document in the target language. We use both Jensen-Shannon divergence and cosine distance. For each document in the query language we rank all documents in the target language and record the rank of the actual translation.

Results averaged over all query/target language pairs are shown in figure 7 for Jensen-Shannon divergence. Cosine-based rankings are significantly worse. It is important to note that the length of documents matters. As noted before, many of the documents in the EuroParl collection consist of short, formulaic sentences. Restricting the query/target pairs to only those with query and target documents that are both longer than 50 words results in significant improvement and reduced variance: the average proportion of query documents for which the true translation is ranked highest goes from 53.9% to 72.7%. Performance continues to improve with longer documents, most likely due to better topic inference. Results vary by language. Table 5 shows results for all target languages with English as a query language. Again, English generally performs better with Romance languages than Germanic languages.

5 Results on Comparable Texts

Directly parallel translations are rare in many languages and can be extremely expensive to produce. However, the growth of the web, and in particular Wikipedia, has made *comparable* text cor-

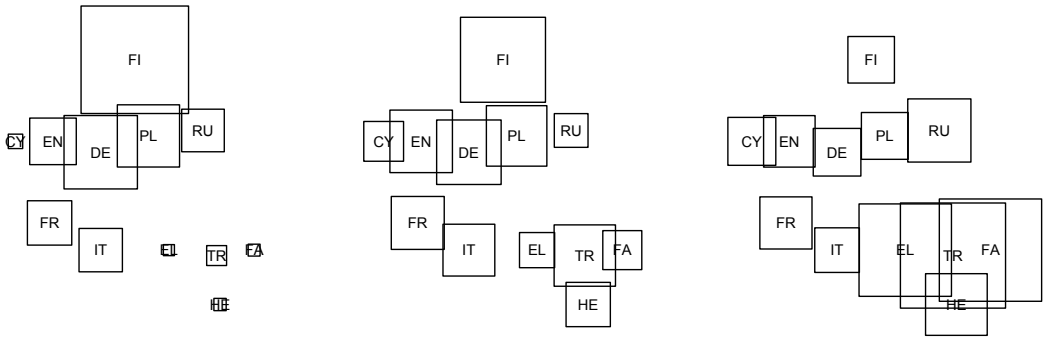


Figure 8: Squares represent the proportion of tokens in each language assigned to a topic. The left topic, *world ski km won*, centers around Nordic counties. The center topic, *actor role television actress*, is relatively uniform. The right topic, *ottoman empire khan byzantine*, is popular in all languages but especially in regions near Istanbul.

Table 5: Percent of English query documents for which the translation was in the top $n \in \{1, 5, 10, 20\}$ documents by JS divergence between topic distributions. To reduce the effect of short documents we consider only document pairs where the query and target documents are longer than 100 words.

Lang	1	5	10	20
DA	78.0	90.7	93.8	95.8
DE	76.6	90.0	93.4	95.5
EL	77.1	90.4	93.3	95.2
ES	81.2	92.3	94.8	96.7
FI	76.7	91.0	94.0	96.3
FR	80.1	91.7	94.3	96.2
IT	79.1	91.2	94.1	96.2
NL	76.6	90.1	93.4	95.5
PT	80.8	92.0	94.7	96.5
SV	80.4	92.1	94.9	96.5

pora – documents that are topically similar but are not direct translations of one another – considerably more abundant than true parallel corpora.

In this section, we explore two questions relating to comparable text corpora and polylingual topic modeling. First, we explore whether comparable document tuples support the alignment of fine-grained topics, as demonstrated earlier using parallel documents. This property is useful for building machine translation systems as well as for human readers who are either learning new languages or analyzing texts in languages they do not know. Second, because comparable texts may not use exactly the same topics, it becomes crucially important to be able to characterize differences in topic prevalence at the document level (do different languages have different perspectives on the same article?) and at the language-wide level (which topics do particular languages focus on?).

5.1 Data Set

We downloaded XML copies of all Wikipedia articles in twelve different languages: Welsh, German, Greek, English, Farsi, Finnish, French, Hebrew, Italian, Polish, Russian and Turkish. These versions of Wikipedia were selected to provide a diverse range of language families, geographic areas, and quantities of text. We preprocessed the data by removing tables, references, images and info-boxes. We dropped all articles in non-English languages that did not link to an English article. In the English version of Wikipedia we dropped all articles that were not linked to by any other language in our set. For efficiency, we truncated each article to the nearest word after 1000 characters and dropped the 50 most common word types in each language. Even with these restrictions, the size of the corpus is 148.5 million words.

We present results for a PLTM with 400 topics. 1000 Gibbs sampling iterations took roughly four days on one CPU with current hardware.

5.2 Which Languages Have High Topic Divergence?

As with EuroParl, we can calculate the Jensen-Shannon divergence between pairs of documents within a comparable document tuple. We can then average over all such document-document divergences for each pair of languages to get an overall "disagreement" score between languages. Interestingly, we find that almost all languages in our corpus, including several pairs that have historically been in conflict, show average JS divergences of between approximately 0.08 and 0.12 for $T = 400$, consistent with our findings for EuroParl translations. Subtle differences of sentiment may be below the granularity of the model.

CY sadwrn blaned gallair at lloeren mytholeg
DE space nasa sojus flug mission
EL διαστημικό sts nasa αγγλ small
EN **space mission launch satellite nasa spacecraft**
FA فضایی ماموریت ناسا مدار فضانورد ماهواره
FI sojuz nasa apollo ensimmäinen space lento
FR spatiale mission orbite mars satellite spatial
HE החלל הארץ חלל כדור א תוכנית
IT spaziale missione programma space sojuz stazione
PL misja kosmicznej stacji misji space nasa
RU космический союз космического спутник станции
TR uzay soyuz ay uzaya salyut sovyetler

CY sbaen madrid el la josé sbaeneg
DE de spanischer spanischen spanien madrid la
EL ισπανίας ισπανία de ισπανός ντε μαδρίτη
EN **de spanish spain la madrid y**
FA اسپانیا اسپانیایی کوبا مادرید de ترین
FI espanja de espanjan madrid la real
FR espagnol espagne madrid espagnole juan y
HE ספרד ספרדית דה מדריד הספרדית קובה
IT de spagna spagnolo spagnola madrid el
PL de hiszpański hiszpanii la juan y
RU де мадрид испании испания испанский de
TR ispanya ispanyol madrid la küba real

CY bardd gerddi iaith beirdd fardd gymraeg
DE dichter schriftsteller literatur gedichte gedicht werk
EL ποιητής ποίηση ποιητή έργο ποιητές ποιήματα
EN **poet poetry literature literary poems poem**
FA شاعر شعر ادبیات فارسی ادبی آثار
FI runoilija kirjailija kirjallisuuden kirjoitti runo julkaisi
FR poète écrivain littérature poésie littéraire ses
HE משורר ספרות שירה סופר שירים המשורר
IT poeta letteratura poesia opere versi poema
PL poeta literatury poezji pisarz in jego
RU поэт его писатель литературы поэзии драматург
TR şair edebiyat şiir yazar edebiyatı adlı

Figure 9: Wikipedia topics (T=400).

Overall, these scores indicate that although individual pages may show disagreement, Wikipedia is on average consistent between languages.

5.3 Are Topics Emphasized Differently Between Languages?

Although we find that if Wikipedia contains an article on a particular subject in some language, the article will tend to be topically similar to the articles about that subject in other languages, we also find that across the whole collection different languages emphasize topics to different extents. To demonstrate the wide variation in topics, we calculated the proportion of tokens in each language assigned to each topic. Figure 8 represents the estimated probabilities of topics given a specific language. Competitive cross-country skiing (left) accounts for a significant proportion of the text in Finnish, but barely exists in Welsh and the languages in the Southeastern region. Meanwhile, interest in actors and actresses (center) is consistent across all languages. Finally, historical topics, such as the Byzantine and Ottoman empires (right) are strong in all languages, but show geographical variation: interest centers around the empires.

6 Conclusions

We introduced a polylingual topic model (PLTM) that discovers topics aligned across multiple languages. We analyzed the characteristics of PLTM in comparison to monolingual LDA, and demonstrated that it is possible to discover aligned topics. We also demonstrated that relatively small numbers of topically comparable document tuples are sufficient to align topics between languages in non-comparable corpora. Additionally, PLTM can support the creation of bilingual lexica for low resource language pairs, providing candidate translations for more computationally intense alignment processes without the sentence-aligned translations typically used in such tasks. When applied to comparable document collections such as Wikipedia, PLTM supports data-driven analysis of differences and similarities across *all* languages for readers who understand *any one* language.

7 Acknowledgments

The authors thank Limin Yao, who was involved in early stages of this project. This work was supported in part by the Center for Intelligent Information Retrieval, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant number IIS-0326249, and in part by Army prime contract number W911NF-07-1-0216 and University of Pennsylvania subaward number 103-548106, and in part by National Science Foundation under NSF grant #CNS-0619337. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

David Blei and Michael Jordan. 2003. Modeling annotated data. In *SIGIR*.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *JMLR*.

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

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *ACL*, pages 771–779.

David Hall, Daniel Jurafsky, and Christopher D. Manning. 2008. Studying the history of ideas using topic models. In *EMNLP*.

Mariko Kawaba, Hiroyuki Nakasaki, Takehito Utsuro, and Tomohiro Fukuhara. 2008. Cross-lingual blog analysis based on multilingual blog distillation from multilingual Wikipedia entries. In *ICWSM*.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of ACL Workshop on Unsupervised Lexical Acquisition*.

Gideon Mann, David Mimno, and Andrew McCallum. 2006. Bibliometric impact measures leveraging topic analysis. In *JCDL*.

Andrew McCallum, Andrés Corrada-Emmanuel, and Xuerui Wang. 2005. Topic and role discovery in social networks. In *IJCAI*.

Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. 2009. Mining multilingual topics from Wikipedia. In *WWW*.

Yik-Cheung Tam, Ian Lane, and Tanja Schultz. 2007. Bilingual LSA-based adaptation for statistical machine translation. *Machine Translation*, 28:187–207.

Hanna Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation methods for topic models. In *ICML*.

Xing Wei and Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In *SIGIR*.

Bing Zhao and Eric P. Xing. 2007. HM-BiTAM: Bilingual topic exploration, word alignment, and translation. In *NIPS*.

Using the Web for Language Independent Spellchecking and Autocorrection

Casey Whitelaw and **Ben Hutchinson** and **Grace Y Chung** and **Gerard Ellis**
Google Inc.
Level 5, 48 Pirrama Rd, Pyrmont NSW 2009, Australia
{whitelaw,benhutch,gracec,ged}@google.com

Abstract

We have designed, implemented and evaluated an end-to-end system spellchecking and autocorrection system that does not require *any* manually annotated training data. The World Wide Web is used as a large noisy corpus from which we infer knowledge about misspellings and word usage. This is used to build an error model and an n-gram language model. A small secondary set of news texts with artificially inserted misspellings are used to tune confidence classifiers. Because no manual annotation is required, our system can easily be instantiated for new languages. When evaluated on human typed data with real misspellings in English and German, our web-based systems outperform baselines which use candidate corrections based on hand-curated dictionaries. Our system achieves 3.8% total error rate in English. We show similar improvements in preliminary results on artificial data for Russian and Arabic.

1 Introduction

Spellchecking is the task of predicting which words in a document are misspelled. These predictions might be presented to a user by underlining the misspelled words. Correction is the task of substituting the well-spelled hypotheses for misspellings. Spellchecking and autocorrection are widely applicable for tasks such as word-processing and postprocessing Optical Character Recognition. We have designed, implemented and evaluated an end-to-end system that performs spellchecking and autocorrection.

The key novelty of our work is that the system was developed entirely without the use of manually annotated resources or any explicitly compiled dictionaries of well-spelled words. Our multi-stage system integrates knowledge from statistical error models and language models (LMs) with a statistical machine learning classifier. At each stage, data are required for training models and determining weights on the classifiers. The models and classifiers are all automatically trained from frequency counts derived from the Web and from news data. System performance has been validated on a set of human typed data. We have also shown that the system can be rapidly ported across languages with very little manual effort.

Most spelling systems today require some handcrafted language-specific resources, such as lexica, lists of misspellings, or rule bases. Systems using statistical models require large annotated corpora of spelling errors for training. Our statistical models require no annotated data. Instead, we rely on the Web as a large noisy corpus in the following ways. 1) We infer information about misspellings from term usage observed on the Web, and use this to build an error model. 2) The most frequently observed terms are taken as a noisy list of potential candidate corrections. 3) Token n-grams are used to build an LM, which we use to make context-appropriate corrections. Because our error model is based on scoring substrings, there is no fixed lexicon of well-spelled words to determine misspellings. Hence, both novel misspelled or well-spelled words are allowable. Moreover, in combination with an n-gram LM component, our system can detect and correct real-word substitutions, ie, word usage and grammatical errors.

Confidence classifiers determine the thresholds for spelling error detection and autocorrection, given error and LM scores. In order to train these classifiers, we require some textual content with some misspellings and corresponding well-spelled words. A small subset of the Web data from news pages are used because we assume they contain

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 890–899,
Singapore, 6-7 August 2009.

relatively few misspellings. We show that confidence classifiers can be adequately trained and tuned without real-world spelling errors, but rather with clean news data injected with artificial misspellings.

This paper will proceed as follows. In Section 2, we survey related prior research. Section 3 describes our approach, and how we use data at each stage of the spelling system. In experiments (Section 4), we first verify our system on data with artificial misspellings. Then we report performance on data with real typing errors in English and German. We also show preliminary results from porting our system to Russian and Arabic.

2 Related Work

Spellchecking and correction are among the oldest text processing problems, and many different solutions have been proposed (Kukich, 1992). Most approaches are based upon the use of one or more manually compiled resources. Like most areas of natural language processing, spelling systems have been increasingly empirical, a trend that our system continues.

The most direct approach is to model the causes of spelling errors directly, and encode them in an algorithm or an error model. Damerau-Levenshtein edit distance was introduced as a way to detect spelling errors (Damerau, 1964). Phonetic indexing algorithms such as Metaphone, used by GNU Aspell (Atkinson, 2009), repesent words by their approximate 'soundslike' pronunciation, and allow correction of words that appear orthographically dissimilar. Metaphone relies upon data files containing phonetic information. Linguistic intuition about the different causes of spelling errors can also be represented explicitly in the spelling system (Deorowicz and Ciura, 2005).

Almost every spelling system to date makes use of a lexicon: a list of terms which are treated as 'well-spelled'. Lexicons are used as a source of corrections, and also to filter words that should be ignored by the system. Using lexicons introduces the distinction between 'non-word' and 'real-word' errors, where the misspelled word is another word in the lexicon. This has led to the two sub-tasks being approached separately (Golding and Schabes, 1996). Lexicon-based approaches have trouble handling terms that do not appear in the lexicon, such as proper nouns, foreign terms, and neologisms, which can account for a large proportion of 'non-dictionary' terms (Ahmad and Kondrak, 2005).

A word's context provides useful evidence as to its correctness. Contextual information can be represented by rules (Mangu and Brill, 1997) or more commonly in an n-gram LM. Mays et al (1991) used a trigram LM and a lexicon, which was shown to be competitive despite only allowing for a single correction per sentence (Wilcox-O'Hearn et al., 2008). Cucerzan and Brill (2004) claim that an LM is much more important than the channel model when correcting Web search queries. In place of an error-free corpus, the Web has been successfully used to correct real-word errors using bigram features (Lapata and Keller, 2004). This work uses pre-defined confusion sets.

The largest step towards an automatically trainable spelling system was the statistical model for spelling errors (Brill and Moore, 2000). This replaces intuition or linguistic knowledge with a training corpus of misspelling errors, which was compiled by hand. This approach has also been extended to incorporate a pronunciation model (Toutanova and Moore, 2002).

There has been recent attention on using Web search query data as a source of training data, and as a target for spelling correction (Yang Zhang and Li, 2007; Cucerzan and Brill, 2004). While query data is a rich source of misspelling information in the form of query-revision pairs, it is not available for general use, and is not used in our approach.

The dependence upon manual resources has created a bottleneck in the development of spelling systems. There have been few language-independent, multi-lingual systems, or even systems for languages other than English. Language-independent systems have been evaluated on Persian (Barari and QasemiZadeh, 2005) and on Arabic and English (Hassan et al., 2008). To our knowledge, there are no previous evaluations of a language-independent system across many languages, for the full spelling correction task, and indeed, there are no pre-existing standard test sets for typed data with real errors and language context.

3 Approach

Our spelling system follows a noisy channel model of spelling errors (Kernighan et al., 1990). For an observed word w and a candidate correction s, we compute $P(s|w)$ as $P(w|s) \times P(s)$.

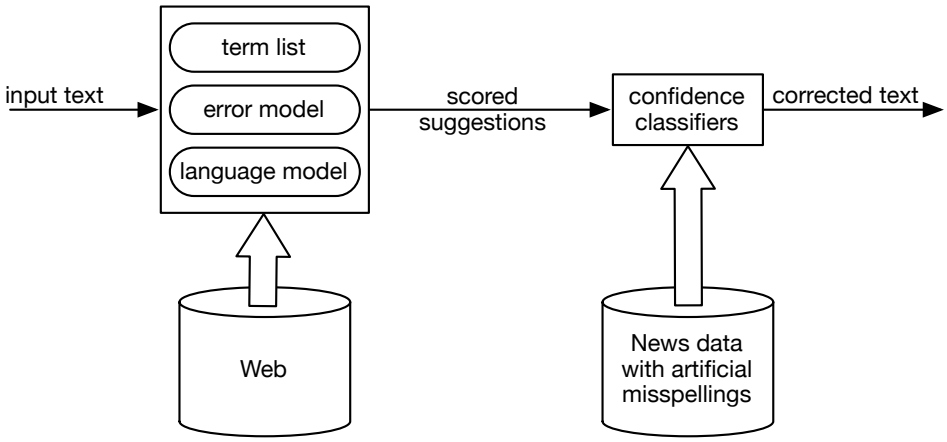


Figure 1: Spelling process, and knowledge sources used.

The text processing workflow and the data used in building the system are outlined in Figure 1 and detailed in this section. For each token in the input text, candidate suggestions are drawn from the term list (Section 3.1), and scored using an error model (Section 3.2). These candidates are evaluated in context using an LM (Section 3.3) and re-ranked. For each token, we use classifiers (Section 3.4) to determine our confidence in whether a word has been misspelled and if so, whether it should be autocorrected to the best-scoring suggestion available.

3.1 Term List

We require a list of terms to use as candidate corrections. Rather than attempt to build a lexicon of words that are well-spelled, we instead take the most frequent tokens observed on the Web. We used a large ($>$ 1 billion) sample of Web pages, tokenized them, and took the most frequently occurring ten million tokens, with very simple filters for non-words (too much punctuation, too short or long). This term list is so large that it should contain most well-spelled words, but also a large number of non-words or misspellings.

3.2 Error Model

We use a substring error model to estimate $P(w|s)$. To derive the error model, let R be a partitioning of s into adjacent substrings, and similarly let T be a partitioning of w, such that $|T| = |R|$. The partitions are thus in one-to-one alignment, and by allowing partitions to be empty, the alignment models insertions and deletions of substrings. Brill and Moore estimate $P(w|s)$ as follows:

$$P(w|s) \approx \max_{R,\, T \text{ s.t. } |T|=|R|} \prod_{i=1}^{|R|} P(T_i|R_i) \qquad (1)$$

Our system restricts partitionings that have substrings of length at most 2.

To train the error model, we require triples of (intended_word, observed_word, count), which are described below. We use maximum likelihood estimates of $P(T_i|R_i)$.

3.2.1 Using the Web to Infer Misspellings

To build the error model, we require as training data a set of (intended_word, observed_word, count) triples, which is compiled from the World Wide Web. Essentially the triples are built by starting with the term list, and a process that automatically discovers, from that list, putative pairs of spelled and misspelled words, along with their counts.

We believe the Web is ideal for compiling this set of triples because with a vast amount of user-generated content, we believe that the Web contains a representative sample of both well-spelled and misspelled text. The triples are not used directly for proposing corrections, and since we have a substring model, they do not need to be an exhaustive list of spelling mistakes.

The procedure for finding and updating counts for these triples also assumes that 1) misspellings tend to be orthographically similar to the intended word; Mays et al (1991) observed that 80% of

misspellings derived from single instances of insertion, deletion, or substitution; and 2) words are usually spelled as intended.

For the error model, we use a large corpus (up to 3.7×10^8 pages) of crawled public Web pages. An automatic language-identification system is used to identify and filter pages for the desired language. As we only require a small window of context, it would also be possible to use an n-gram collection such as the Google Web 1T dataset.

Finding Close Words. For each term in the term list (defined in Section 3.1), we find all other terms in the list that are "close" to it. We define closeness using Levenshtein-Damerau edit distance, with a conservative upper bound that increases with word length (one edit for words of up to four characters, two edits for up to twelve characters, and three for longer words). We compile the term list into a trie-based data structure which allows for efficient searching for all terms within a maximum edit distance. The computation is 'embarassingly parallel' and hence easily distributable. In practice, we find that this stage takes tens to hundreds of CPU-hours.

Filtering Triples. At this stage, for each term we have a cluster of orthographically similar terms, which we posit are potential misspellings. The set of pairs is reflexive and symmetric, e.g. it contains both (*recieve*, *receive*) and (*receive*, *recieve*). The pairs will also include e.g. (*deceive*, *receive*). On the assumption that words are spelled correctly more often than they are misspelled, we next filter the set such that the first term's frequency is at least 10 times that of the second term. This ratio was chosen as a conservative heuristic filter.

Using Language Context. Finally, we use the contexts in which a term occurs to gather directional weightings for misspellings. Consider a term w; from our source corpus, we collect the set of contexts $\{c_i\}$ in which w occurs. The definition of a context is relatively arbitrary; we chose to use a single word on each side, discarding contexts with fewer than a total of ten observed occurrences. For each context c_i, candidate "intended" terms are w and w's close terms (which are at least 10 times as frequent as w). The candidate which appears in context c_i the most number of times is deemed to be the term intended by the user in that context.

The resulting dataset consists of triples of the original observed term, one of the "intended" terms as determined by the above algorithm, and the number of times this term was intended. For a single term, it is possible (and common) to have multiple possible triples, due to the context-based assignment.

Inspecting the output of this training process shows some interesting patterns. Overall, the dataset is still noisy; there are many instances where an obviously misspelled word is not assigned a correction, or only some of its instances are. The dataset contains around 100 million triples, orders of magnitude larger than any manually compiled list of misspellings . The kinds of errors captured in the dataset include stereotypical spelling errors, such as *acomodation*, but also OCR-style errors. *computationaUy* was detected as a misspelling of *computationally* where the 'U' is an OCR error for 'll'; similarly, *Postmodem* was detected as a misspelling of *Postmodern* (an example of 'keming').

The data also includes examples of 'real-word' errors. For example, 13% of occurrences of *occidental* are considered misspellings of *accidental*; contrasting with 89% of occurrences of the non-word *accidential*. There are many examples of terms that would not be in a normal lexicon, including neologisms (*mulitplayer* for *multiplayer*), companies and products (*Playstaton* for *Playstation*), proper nouns (*Schwarznegger* for *Schwarzenegger*) and internet domain names (*mysapce.com* for *myspace.com*).

3.3 Language Model

We estimate $P(s)$ using n-gram LMs trained on data from the Web, using Stupid Backoff (Brants et al., 2007). We use both forward and backward context, when available. Contrary to Brill and Moore (2000), we observe that user edits often have both left and right context, when editing a document.

When combining the error model scores with the LM scores, we weight the latter by taking their λ'th power, that is

$$P(w|s) * P(s)^{\lambda} \quad (2)$$

The parameter λ reflects the relative degrees to which the LM and the error model should be trusted. The parameter λ also plays the additional role of correcting our error model's misestimation of the rate at which people make errors. For example, if errors are common then by increasing λ we

can reduce the value of $P(w|w) * P(w)^{\lambda}$ relative to $\sum_{s \neq w} P(s|w) * P(s)^{\lambda}$.

We train λ by optimizing the average inverse rank of the correct word on our training corpus, where the rank is calculated over all suggestions that we have for each token.

During initial experimentation, it was noticed that our system predicted many spurious autocorrections at the beginnings and ends of sentences (or in the case of sentence fragments, the end of the fragment). We hypothesized that we were weighting the LM scores too highly in such cases. We therefore conditioned λ on how much context was available, obtaining values $\lambda_{i,j}$ where i, j represent the amount of context available to the LM to the left and right of the current word. i and j are capped at n, the order of the LM.

While conditioning λ in this way might at first appear ad hoc, it has a natural interpretation in terms of our confidence in the LM. When there is no context to either side of a word, the LM simply uses unigram probabilities, and this is a less trustworthy signal than when more context is available.

To train $\lambda_{i,j}$ we partition our data into bins corresponding to pairs i, j and optimize each $\lambda_{i,j}$ independently.

Training a constant λ, a value of 5.77 was obtained. The conditioned weights $\lambda_{i,j}$ increased with the values of i and j, ranging from $\lambda_{0,0} = 0.82$ to $\lambda_{4,4} = 6.89$. This confirmed our hypothesis that the greater the available context the more confident our system should be in using the LM scores.

3.4 Confidence Classifiers for Checking and Correction

Spellchecking and autocorrection were implemented as a three stage process. These employ confidence classifiers whereby precision-recall tradeoffs could be tuned to desirable levels for both spellchecking and autocorrection.

First, all suggestions s for a word w are ranked according to their $P(s|w)$ scores. Second, a spellchecking classifier is used to predict whether w is misspelled. Third, if w is both predicted to be misspelled and s is non-empty, an autocorrection classifier is used to predict whether the top-ranked suggestion is correct.

The spellchecking classifier is implemented using two embedded classifiers, one of which is used when s is empty, and the other when it is non-empty. This design was chosen because the useful signals for predicting whether a word is misspelled might be quite different when there are no suggestions available, and because certain features are only applicable when there are suggestions.

Our experiments will compare two classifier types. Both rely on training data to determine threshold values and training weights.

A "simple" classifier which compares the value of $log(P(s|w)) - log(P(w|w))$, for the original word w and the top-ranked suggestion s, with a threshold value. If there are no suggestions other than w, then the $log(P(s|w))$ term is ignored.

A logistic regression classifier that uses five feature sets. The first set is a scores feature that combines the following scoring information (i) $log(P(s|w)) - log(P(w|w))$ for top-ranked suggestion s. (ii) LM score difference between the original word w and the top suggestion s. (iii) $log(P(s|w)) - log(P(w|w))$ for second top-ranked suggestion s. (iv) LM score difference between w and second top-ranked s. The other four feature sets encode information about case signatures, number of suggestions available, the token length, and the amount of left and right context.

Certain categories of tokens are blacklisted, and so never predicted to be misspelled. These are numbers, punctuation and symbols, and single-character tokens.

The training process has three stages. (1) The context score weighting is trained, as described in Section 3.3. (2) The spellchecking classifier is trained, and tuned on held-out development data. (3) The autocorrection classifier is trained on the instances with suggestions that the spellchecking classifier predicts to be misspelled, and it too is tuned on held-out development data.

In the experiments reported in this paper, we trained classifiers so as to maximize the F_1-score on the development data. We note that the desired behaviour of the spellchecking and autocorrection classifiers will differ depending upon the application, and that it is a strength of our system that these can be tuned independently.

3.4.1 Training Using Artificial Data

Training and tuning the confidence classifiers require supervised data, in the form of pairs of misspelled and well-spelled documents. And indeed we posit that relatively noiseless data are needed to train robust classifiers. Since these data are

Language	Sentences	
	Train	Test
English	116k	58k
German	87k	44k
Arabic	8k	4k
Russian	8k	4k

Table 1: Artificial data set sizes. The development set is approximately the same size as the training set.

not generally available, we instead use a clean corpus into which we artificially introduce misspellings. While this data is not ideal, we show that in practice it is sufficient, and removes the need for manually-annotated gold-standard data.

We chose data from news pages crawled from the Web as the original, well-spelled documents. We chose news pages as an easily identifiable source of text which we assume is almost entirely well-spelled. Any source of clean text could be used. For each language the news data were divided into three non-overlapping data sets: the training and development sets were used for training and tuning the confidence classifiers, and a test set was used to report evaluation results. The data set sizes, for the languages used in this paper, are summarized in Table 1.

Misspelled documents were created by artificially introducing misspelling errors into the well-spelled text. For all data sets, spelling errors were randomly inserted at an average rate of 2 per hundred characters, resulting in an average word misspelling rate of 9.2%. With equal likelihood, errors were either character deletions, transpositions, or insertions of randomly selected characters from within the same document.

4 Experiments

4.1 Typed Data with Real Errors

In the absence of user data from a real application, we attempted our initial evaluation with typed data via a data collection process. Typed data with real errors produced by humans were collected. We recruited subjects from our coworkers, and asked them to use an online tool customized for data collection. Subjects were asked to randomly select a Wikipedia article, copy and paste several text-only paragraphs into a form, and retype those paragraphs into a subsequent form field. The subjects were asked to pick an article about a favorite city or town. The subjects were asked to type at a normal pace avoiding the use of backspace or delete buttons. The data were tokenized, automatically segmented into sentences, and manually preprocessed to remove certain gross typing errors. For instance, if the typist omitted entire phrases/sentences by mistake, the sentence was removed. We collected data for English from 25 subjects, resulting in a test set of 11.6k tokens, and 495 sentences. There were 1251 misspelled tokens (10.8% misspelling rate.)

Data were collected for German Wikipedia articles. We asked 5 coworkers who were German native speakers to each select a German article about a favorite city or town, and use the same online tool to input their typing. For some typists who used English keyboards, they typed ASCII equivalents to non-ASCII characters in the articles. This was accounted for in the preprocessing of the articles to prevent misalignment. Our German test set contains 118 sentences, 2306 tokens with 288 misspelled tokens (12.5% misspelling rate.)

4.2 System Configurations

We compare several system configurations to investigate each component's contribution.

4.2.1 Baseline Systems Using Aspell

Systems 1 to 4 have been implemented as baselines. These use GNU Aspell, an open source spell checker (Atkinson, 2009), as a suggester component plugged into our system instead of our own Web-based suggester. Thus, with Aspell, the suggestions and error scores proposed by the system would all derive from Aspell's handcrafted custom dictionary and error model. (We report results using the best combination of Aspell's parameters that we found.)

System 1 uses Aspell tuned with the logistic regression classifier. System 2 adds a context-weighted LM, as per Section 3.3, and uses the "simple" classifier described in Section 3.4. System 3 replaces the simple classifier with the logistic regression classifier. System 4 is the same but does not perform blacklisting.

4.2.2 Systems Using Web-based Suggestions

The Web-based suggester proposes suggestions and error scores from among the ten million most frequent terms on the Web. It suggests the 20 terms with the highest values of $P(w|s) \times f(s)$ using the Web-derived error model.

Systems 5 to 8 correspond with Systems 1 to 4, but use the Web-based suggestions instead of Aspell.

4.3 Evaluation Metrics

In our evaluation, we aimed to select metrics that we hypothesize would correlate well with real performance in a word-processing application. In our intended system, misspelled words are autocorrected when confidence is high and misspelled words are flagged when a highly confident suggestion is absent. This could be cast as a simple classification or retrieval task (Reynaert, 2008), where traditional measures of precision, recall and F metrics are used. However we wanted to focus on metrics that reflect the quality of end-to-end behavior, that account for the combined effects of flagging and automatic correction. Essentially, there are three states: a word could be unchanged, flagged or corrected to a suggested word. Hence, we report on error rates that measure the errors that a user would encounter if the spellchecking/autocorrection were deployed in a word-processor. We have identified 5 types of errors that a system could produce:

1. E_1: A misspelled word is wrongly corrected.
2. E_2: A misspelled word is not corrected but is flagged.
3. E_3: A misspelled word is not corrected or flagged.
4. E_4: A well spelled word is wrongly corrected.
5. E_5: A well spelled word is wrongly flagged.

It can be argued that these errors have varying impact on user experience. For instance, a well spelled word that is wrongly corrected is more frustrating than a misspelled word that is not corrected but is flagged. However, in this paper, we treat each error equally.

E_1, E_2, E_3 and E_4 pertain to the correction task. Hence we can define Correction Error Rate (CER):

$$\text{CER} = \frac{E_1 + E_2 + E_3 + E_4}{T}$$

where T is the total number of tokens. E_3 and E_5 pertain to the nature of flagging. We define Flagging Error Rate (FER) and Total Error Rate (TER):

$$\text{FER} = \frac{E_3 + E_5}{T}$$

$$\text{TER} = \frac{E_1 + E_2 + E_3 + E_4 + E_5}{T}$$

For each system, we computed a No Good Suggestion Rate (NGS) which represents the proportion of misspelled words for which the suggestions list did not contain the correct word.

5 Results and Discussion

5.1 Experiments with Artificial Errors

System	TER	CER	FER	NGS
1. Aspell, no LM, LR	17.65	6.38	12.35	18.3
2. Aspell, LM, Sim	4.82	2.98	2.86	18.3
3. Aspell, LM, LR	4.83	2.87	2.84	18.3
4. Aspell, LM, LR (no blacklist)	22.23	2.79	19.89	16.3
5. WS, no LM, LR	9.06	7.64	6.09	10.1
6. WS, LM, Sim	2.62	2.26	1.43	10.1
7. WS, LM, LR	**2.55**	**2.21**	**1.29**	10.1
8. WS, LM, LR (no blacklist)	21.48	2.21	19.75	8.9

Table 2: Results for English news data on an independent test set with artificial spelling errors. Numbers are given in percentages. LM: Language Model, Sim: Simple, LR: Logistic Regression, WS: Web-based suggestions. NGS: No good suggestion rate.

Results on English news data with artificial spelling errors are displayed in Table 2. The systems which do not employ the LM scores perform substantially poorer that the ones with LM scores. The Aspell system yields a total error rate of 17.65% and our system with Web-based suggestions yields TER of 9.06%.

When comparing the simple scorer with the logistic regression classifier, the Aspell Systems 2 and 3 generate similar performances while the confidence classifier afforded some gains in our Web-based suggestions system, with total error reduced from 2.62% to 2.55%. The ability to tune each phase during development has so far proven more useful than the specific features or classifier used. Blacklisting is crucial as seen by our results for Systems 4 and 8. When the blacklisting mechanism is not used, performance steeply declines.

When comparing overall performance for the data between the Aspell systems and the Web-based suggestions systems, our Web-based suggestions fare better across the board for the news data with artificial misspellings. Performance

gains are evident for each error metric that was examined. Total error rate for our best system (System 7) reduces the error of the best Aspell system (System 3) by 45.7% (from 4.83% to 2.62%). In addition, our no good suggestion rate is only 10% compared to 18% in the Aspell system. Even where no LM scores are used, our Web-based suggestions system outperforms the Aspell system.

The above results suggest that the Web-based suggestions system performs at least as well as the Aspell system. However, it must be highlighted that results on the test set with artificial errors does not guarantee similar performance on real user data. The artificial errors were generated at a systematically uniform rate, and are not modeled after real human errors made in real word-processing applications. We attempt to consider the impact of real human errors on our systems in the next section.

5.2 Experiments with Human Errors

System	TER	CER	FER	NGS
English Aspell	4.58	**3.33**	2.86	23.0
English WS	**3.80**	3.41	**2.24**	17.2
German Aspell	14.09	10.23	5.94	44.4
German WS	**9.80**	**7.89**	**4.55**	32.3

Table 3: Results for Data with Real Errors in English and German.

Results for our system evaluated on data with real misspellings in English and in German are shown in Table 3. We used the systems that performed best on the artificial data (System 3 for Aspell, and System 7 for Web suggestions). The misspelling error rates of the test sets were 10.8% and 12.5% respectively, higher than those of the artificial data which were used during development. For English, the Web-based suggestions resulted in a 17% improvement (from 4.58% to 3.80%) in total error rate, but the correction error rate was slightly (2.4%) higher.

By contrast, in German our system improved total error by 30%, from 14.09% to 9.80%. Correction error rate was also much lower in our German system, comparing 7.89% with 10.23% for the Aspell system. The no good suggestion rates for the real misspelling data are also higher than that of the news data. Our suggestions are limited to an edit distance of 2 with the original, and it was found that in real human errors, the average edit distance of misspelled words is 1.38 but for our small data, the maximum edit distance is 4 in English and 7 in German. Nonetheless, our no good suggestion rates (17.2% and 32.3%) are much lower than those of the Aspell system (23% and 44%), highlighting the advantage of not using a hand-crafted lexicon.

Our results on real typed data were slightly worse than those for the news data. Several factors may account for this. (1) While the news data test set does not overlap with the classifier training set, the nature of the content is similar to the train and dev sets in that they are all news articles from a one week period. This differs substantially from Wikipedia article topics that were generally about the history and sights a city. (2) Second, the method for inserting character errors (random generation) was the same for the news data sets while the real typed test set differed from the artificial errors in the training set. Typed errors are less consistent and error rates differed across subjects. More in depth study is needed to understand the nature of real typed errors.

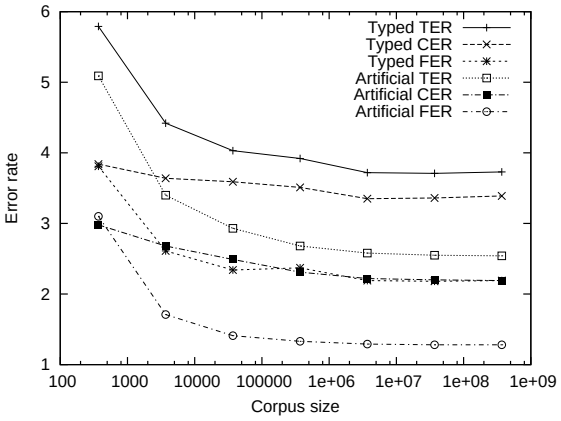


Figure 2: Effect of corpus size used to train the error model.

5.3 Effect of Web Corpus Size

To determine the effects of the corpus size on our automated training, we evaluated System 7 using error models trained on different corpus sizes. We used corpora containing $10^3, 10^4, \ldots, 10^9$ Web pages. We evaluated on the data set with real errors. On average, about 37% of the pages in our corpus were in English. So the number of pages we used ranged from about 370 to about 3.7×10^8. As shown in Figure 2, the gains are small after about 10^6 documents.

5.4 Correlation across data sets

We wanted to establish that performance improvement on the news data with artificial errors are likely to lead to improvement on typed data with real errors. The seventeen English systems reported in Table 3, Table 2 and Figure 2 were each evaluated on both English test sets. The rank correlation coefficient between total error rates on the two data sets was high ($\tau = 0.92$; $p < 5 \times 10^{-6}$). That is, if one system performs better than another on our artificial spelling errors, then the first system is very likely to also perform better on real typing errors.

5.5 Experiments with More Languages

System	TER	CER	FER	NGS
German Aspell	8.64	4.28	5.25	29.4
German WS	4.62	3.35	2.27	16.5
Arabic Aspell	11.67	4.66	8.51	25.3
Arabic WS	4.64	3.97	2.30	15.9
Russian Aspell	16.75	4.40	13.11	40.5
Russian WS	3.53	2.45	1.93	15.2

Table 4: Results for German, Russian, Arabic news data.

Our system can be trained on many languages with almost no manual effort. Results for German, Arabic and Russian news data are shown in Table 4. Performance improvements by the Web suggester over Aspell are greater for these languages than for English. Relative performance improvements in total error rates are 47% in German, 60% in Arabic and 79% in Russian. Differences in no good suggestion rates are also very pronounced between Aspell and the Web suggester.

It cannot be assumed that the Arabic and Russian systems would perform as well on real data. However the correlation between data sets reported in Section 5.4 lead us to hypothesize that a comparison between the Web suggester and Aspell on real data would be favourable.

6 Conclusions

We have implemented a spellchecking and autocorrection system and evaluated it on typed data. The main contribution of our work is that while this system incorporates several knowledge sources, an error model, LM and confidence classifiers, it does not require any manually annotated resources, and infers its linguistic knowledge entirely from the Web. Our approach begins with a very large term list that is noisy, containing both spelled and misspelled words, and derived automatically with no human checking for whether words are valid or not.

We believe this is the first published system to obviate the need for any hand labeled data. We have shown that system performance improves from a system that embeds handcrafted knowledge, yielding a 3.8% total error rate on human typed data that originally had a 10.8% error rate. News data with artificially inserted spellings were sufficient to train confidence classifiers to a satisfactory level. This was shown for both German and English. These innovations enable the rapid development of a spellchecking and correction system for any language for which tokenizers exist and string edit distances make sense. We have done so for Arabic and Russian.

In this paper, our results were obtained without any optimization of the parameters used in the process of gathering data from the Web. We wanted to minimize manual tweaking particularly if it were necessary for every language. Thus heuristics such as the number of terms in the term list, the criteria for filtering triples, and the edit distance for defining close words were crude, and could easily be improved upon. It may be beneficial to perform more tuning in future. Furthermore, future work will involve evaluating the performance of the system for these language on real typed data.

7 Acknowledgment

We would like to thank the anonymous reviewers for their useful feedback and suggestions. We also thank our colleagues who participated in the data collection.

References

Farooq Ahmad and Grzegorz Kondrak. 2005. Learning a spelling error model from search query logs. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 955–962, Morristown, NJ, USA. Association for Computational Linguistics.

K. Atkinson. 2009. Gnu aspell. In *Available at* http://aspell.net.

Loghman Barari and Behrang QasemiZadeh. 2005. Clonizer spell checker adaptive, language independent spell checker. In Ashraf Aboshosha et al., editor, *Proc. of the first ICGST International Confer-*

ence on Artificial Intelligence and Machine Learning AIML 05, volume 05, pages 65–71, Cairo, Egypt, Dec. ICGST, ICGST.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics.

S. Cucerzan and E. Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP 2004*, pages 293–300.

F.J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM 7*, pages 171–176.

S. Deorowicz and M.G. Ciura. 2005. Correcting spelling errors by modelling their causes. *International Journal of Applied Mathematics and Computer Science*, 15(2):275–285.

Andrew R. Golding and Yves Schabes. 1996. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 71–78.

Ahmed Hassan, Sara Noeman, and Hany Hassan. 2008. Language independent text correction using finite state automata. In *Proceedings of the 2008 International Joint Conference on Natural Language Processing (IJCNLP, 2008)*.

Mark D. Kernighan, Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th conference on Computational linguistics*, pages 205–210. Association for Computational Linguistics.

K. Kukich. 1992. Techniques for automatically correcting words in texts. *ACM Computing Surveys 24*, pages 377–439.

Mirella Lapata and Frank Keller. 2004. The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of nlp tasks. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 121–128, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Lidia Mangu and Eric Brill. 1997. Automatic rule acquisition for spelling correction. In Douglas H. Fisher, editor, *ICML*, pages 187–194. Morgan Kaufmann.

Eric Mays, Fred J. Damerau, and Robert L. Mercer. 1991. Context based spelling correction. *Information Processing and Management*, 27(5):517.

M.W.C. Reynaert. 2008. All, and only, the errors: More complete and consistent spelling and ocr-error correction evaluation. In *Proceedings of the sixth international language resources and evaluation.*

Kristina Toutanova and Robert Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 144–151.

L. Amber Wilcox-O'Hearn, Graeme Hirst, and Alexander Budanitsky. 2008. Real-word spelling correction with trigrams: A reconsideration of the mays, damerau, and mercer model. In Alexander F. Gelbukh, editor, *CICLing*, volume 4919 of *Lecture Notes in Computer Science*, pages 605–616. Springer.

Wei Xiang Yang Zhang, Pilian He and Mu Li. 2007. Discriminative reranking for spelling correction. In *The 20th Pacific Asia Conference on Language, Information and Computation*.

Statistical Estimation of Word Acquisition with Application to Readability Prediction

Paul Kidwell
Department of Statistics
Purdue University
West Lafayette, IN
kidwellpaul@gmail.com

Guy Lebanon
College of Computing
Georgia Institute of Technology
Atlanta, GA
lebanon@cc.gatech.edu

Kevyn Collins-Thompson
Microsoft Research
Redmond, WA
kevynct@microsoft.com

Abstract

Models of language learning play a central role in a wide range of applications: from psycholinguistic theories of how people acquire new word knowledge, to information systems that can automatically match content to users' reading ability. We present a novel statistical approach that can infer the distribution of a word's likely acquisition age automatically from authentic texts collected from the Web. We then show that combining these acquisition age distributions for all words in a document provides an effective semantic component for predicting reading difficulty of new texts. We also compare our automatically inferred acquisition ages with norms from existing oral studies, revealing interesting historical trends as well as differences between oral and written word acquisition processes.

1 Introduction

Word acquisition refers to the temporal process by which children learn the meaning and understanding of new words. Some words are acquired at a very early age, some are acquired at early primary school grades, and some are acquired at high school or even later in life as the individual undergoes experiences related to that word. A related concept to acquisition age is document grade level readability which refers to the school grade level of the document's intended audience. It applies in situations where documents are written with the expressed intent of being understood by children in a certain school grade. For example, textbooks authored specifically for fourth graders are said to have readability grade level four.

We develop and evaluate a novel statistical model that draws a connection between document grade level readability and age acquisition distributions. Based on previous work in the area, we define a model for document readability using a logistic Rasch model and the quantiles of the acquisition age distributions. We then proceed to infer the age acquisition distributions for different words from document readability data collected by crawling the web.

We examine the inferred acquisition distributions from two perspectives. First, we analyze and contrast them with previous studies on oral word acquisition, revealing interesting historical trends as well as differences between oral and written word acquisition processes. Second, the inferred acquisition distributions serve as parameters for the readability model, which enables us to predict the readability level of novel documents.

To our knowledge, this is the first published study of a method to 'reverse-engineer' individual word acquisition statistics from graded texts. By obtaining such a fine-grained model of how language evolves over time, we obtain a new, rich source of semantic features for a document. The increasing amounts of content available from the Web and other sources also means that these flexible models of authentic usage can be easily adapted for different tasks and populations. Our work serves to complement the growing body of research using statistics and machine learning for language learning tasks, and has applications including predicting reading difficulty for Web pages and other non-traditional documents, reader-specific example and question generation for lexical practice in intelligent tutoring systems, and analysis tools for language learning research.

2 A Model for Document Readability and Word Acquisition

For a fixed word and a fixed population of individuals $\mathcal{T}$ the age of acquisition (AoA) distribution p_w represents the age at which word w was

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 900–909,
Singapore, 6-7 August 2009.

acquired by the population. Existing AoA norm studies almost universally summarize AoA ratings in terms of two parameters: mean and standard deviation, ignoring higher-level moments such as skew. For direct comparison with these studies we follow this convention and thus our goal is to estimate AoA for a word w in terms of mean μ_w and standard deviation σ_w parameters using the (truncated) normal distribution

$$p_w(t) \propto N(t\,;\mu_w,\sigma_w) = \frac{e^{-(t-\mu_w)^2/(2\sigma_w^2)}}{\sqrt{2\pi\sigma_w^2}} \quad (1)$$

where the proportionality constant ensures that the distribution is normalized over the range of ages under consideration e.g., $t \in [6, 18]$ for school grades. It is important to note that our model is not restricted by the assumption of (1) and can be readily extended to the Gamma family of distributions, if modeling asymmetric spread in the distribution is appropriate.

For a fixed vocabulary V of distinct words the age acquisition distributions for all words $w \in V$ are defined using $2|V|$ parameters

$$\{(\mu_w, \sigma_w) : w \in V\}. \quad (2)$$

These parameters, which are the main objects of interest, can in principle be estimated from data using standard statistical techniques. Unfortunately, data containing explicit acquisition ages is very difficult to obtain reliably. Explicit word acquisition data is based on interviewing adults regarding their age acquisition process during childhood and so may be unreliable and difficult to obtain for a large representative group of people.

On the other hand, it is possible to reliably collect large quantities of readability data defined as pairs of documents and ages of intended audience. As we demonstrate later in the paper, such data may be automatically obtained by crawling specialized resources on the Web. We demonstrate how to use such data to estimate the word acquisition parameters (2) and to use the estimates to predict future readability ages.

Traditionally, document readability has been defined in terms of the school grade level at which a large portion of the words have been acquired by most children (Chall and Dale, 1995). We propose the following interpretation of that definition, which is made appropriate for quantitative studies by taking into account the inherent randomness in the acquisition process.

Definition 1. *A document $d = (w_1, \ldots, w_m)$ is said to have $(1-\epsilon_1, 1-\epsilon_2)$-readability level t if by age t no less than $1-\epsilon_1$ percent of the words in d have been acquired each by no less than $1-\epsilon_2$ percent of the population.*

We denote by q_w the quantile function of the cdf corresponding to the acquisition distribution p_w. In other words, $q_w(r)$ represents the age at which r percent of the population $\mathcal{T}$ have acquired word w. Despite the fact that it does not have a closed form, it is a continuous and smooth function of the parameters μ_w, σ_w in (1) (assuming $\mathcal{T}$ is infinite) and can be tabulated before inference begins.

Following Definition 1 we define a logistic Rasch readability model:

$$\log \frac{P(d \text{ is } (s,r)\text{-readable at age } t)}{1 - P(d \text{ is } (s,r)\text{-readable at age } t)} = \theta(q_d(s,r) - t) \quad (3)$$

where $q_d(s,r)$ is the s quantile of $\{q_{w_i}(r) : i = 1, \ldots, m\}$. An equivalent formulation to (3) that makes the probability model more explicit is

$$P(d \text{ is } (s,r)\text{-readable at age } t) = \frac{\exp(\theta(q_d(s,r) - t))}{1 + \exp(\theta(q_d(s,r) - t))}. \quad (4)$$

In other words, the probability of a document d being (s,r)-readable increases exponentially with $q_d(s,r)$ which is the age at which s percent of the words in d have been acquired each by r percent of the population.

The parameter $r = 1 - \epsilon_2$ determines what it means for a word to be acquired and is typically considered to be a high value such as 0.8. The parameter $s = 1 - \epsilon_1$ determines how many of the document words need to be acquired for it to be readable. It can be set to a high value such as 0.9 if a very precise understanding is required for readability but can be reduced when a more modest definition of readability applies.

We note that due to the discreteness of the set $\{q_{w_i}(r) : i = 1, \ldots, m\}$, neither $q_d(s,r)$ nor the loglikelihood are differentiable in the parameters (2). This raises some practical difficulties with respect to the computational maximization of the likelihood and subsequent estimation of (2). However, for long documents containing a large number of words, $q_d(s,r)$ is approximately smooth which motivates a maximum likelihood procedure using gradient descent on a smoothed version of

$q_d(s)$. Alternative optimization techniques which do not require smoothness may also be used.

In the case of a normal distribution (1) we have that a word is acquired by r percent of the population at age $w = \mu + \Phi^{-1}(r)\sigma$, where Φ is the cumulative distribution function (cdf) of the normal distribution. To investigate the distribution of acquisition ages we assume that the μ, σ parameters corresponding to different words in a document are drawn from Gamma distributions $\mu \sim G(\alpha_1, \beta_1)$ and $\sigma \sim G(\alpha_2, \beta_2)$. The normal and Gamma distributions are chosen in part because they are flexible enough to model many situations and also admit good statistical estimation theory. Noting that $\Phi^{-1}(r)\sigma \sim G(\alpha_2, \Phi^{-1}(r)\beta_2)$, we can write the distribution of the acquisition ages as the following convolution

$$f_W(w) = \frac{w^{\alpha_1+\alpha_2-1}e^{-w/\beta_2}}{\Gamma(\alpha_1)\Gamma(\alpha_2)\beta_1^{\alpha_1}\beta_2^{\alpha_2}} * \int_0^1 \frac{t^{\alpha_1-1}e^{\frac{(\beta_1-\beta_2)tw}{\beta_1\beta_2}}}{(1-t)^{1-\alpha_2}}dt$$

which reverts to a Gamma when $\beta_1 = \beta_2$.

The distribution of the s-percentile of f_W, which amounts to (r, s)-readability of documents, can be analyzed by combining f_W above with a standard normal approximation of order statistics (e.g., (David and Nagaraja, 2003))

$$X_{\lfloor mp \rfloor} \sim N\left(F_W^{-1}(p), \frac{p(1-p)}{m[f_W(F_W^{-1}(p))]^2}\right)$$

where m is the document length and F_W is the cdf corresponding to f_W.

Figure 1 shows the relationship between document length and confidence interval (CI) width in readability prediction. It contrasts the CI widths for model based intervals and empirical intervals. In both cases, documents of lengths larger than 100 words provide CI widths shorter than 1 year. This finding is also noteworthy as it provides empirical support for the long-standing 'rule-of-thumb' that readability measures become unreliable for passages of less than 100 words (Fry, 1990).

3 Experimental Results

Our experimental study is divided into three parts. The first part examines the word acquisition distributions that were estimated based on readability data. The second part compares the estimated

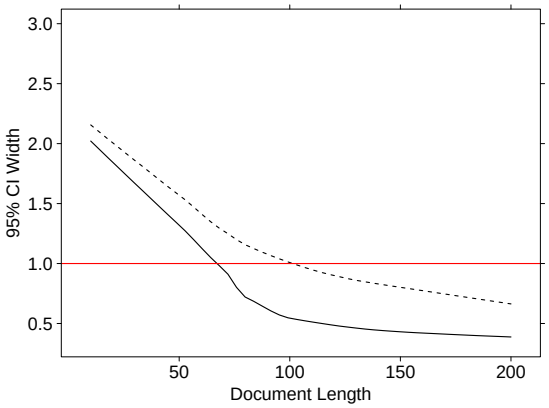


Figure 1: A comparison of model (dashed) vs. empirical (solid) 95% confidence interval widths as a function of document length ($r = 0.9$ and $s = 0.7$). CI widths were computed using 1000 Monte Carlo samples generated from the f_W model fit to the data and from the empirical distribution. Word distributions correspond to a 1577 word document written for a 7th grade audience taken from the Web 1-12 corpus.

(written) acquisition ages with oral acquisition ages obtained from interview studies reported in the literature. The third part focuses on using the estimated word acquisition distributions to predict document readability. These three experimental studies are described in the three subsections below.

In our experiments we used three readability datasets. The corpora were compiled by crawling web pages containing documents authored for audiences of specific grade levels. The Web 1-12 data contains 373 documents, with each document written for a particular school grade level in the range 1-12. The Weekly Reader (WR) dataset, was obtained by crawling the commercial website www.wrtoolkit.com after receiving special permission. That dataset contains a total of 1780 documents, with 4 readability levels ranging from 2 to 5 indicating the school grade levels of the intended audience. A total of 788 documents with readability between grades 2 and 5 and having length greater than 50 words were selected from 1780 documents. The Reading A-Z dataset, contains a set of 215 documents was obtained from Reading A-Z.com, spanning grade 1 through grade 6.

The grade levels in these three corpora, which correspond to US school grades, were either explicitly specified by the organization or authors

who created the text, or implicit in the classroom curriculum page where the document was acquired. The pages were drawn from a wide range of subject areas, including history, science, geography, and fiction.

To reduce the possibility of overfitting, we used a common feature selection technique of eliminating words appearing in less than 4 documents. In the experiments we used maximum likelihood to estimate the model parameters $\{(\mu_w, \sigma_w^2) : w \in V\}$ for the Rasch model (3). The maximum likelihood was obtained using a non-smooth coordinate descent procedure.

3.1 Estimation of Word Acquisition Distributions

Figure 2 displays the inferred age acquisition distributions and empirical word appearances of three words: `thought` (left), `multitude` (middle), and `assimilate` (right). In these plots, the empirical cdf of word appearances is indicated by a piecewise constant line while the probability density function of the estimated AoA distribution is indicated by a dashed line. The vertical line indicates the 0.8 quantile of the AoA distribution which corresponds to the grade by which 80% of the children have acquired the word.

The word `assimilation` appears in 2 documents having 12th grade readability. The high grade level of these documents results in a high estimated acquisition age and the paucity of observations leads to a large uncertainty in this estimate as seen by the variance of the acquisition age distribution. The word `thought` appears several times in multiple grades. It is first observed in the 1st grade and not again until the 4th grade resulting in an estimated acquisition age falling between the two. The variance of this acquisition distribution is relatively small due to the frequent use of this word. The empirical cdf shows that `multitude` is used in grades 6, 8, and 9. Relative to `thought` and `assimilation` the word `multitude` was used less and more frequently respectively, which leads to an acquisition age distribution with a larger variance than that of `thought` and smaller than that of `assimilation`.

The relationship in Figure 2 between the empirical word appearances and the age acquisition distribution demonstrates the following behavior: (a) The variance of the age acquisition distribution goes down as the word appears in more documents, and (b) the mean of the AoA distribution tends to be lower than the mean of the empirical word appearance distribution, and in many cases even smaller than the first grade in which the word appeared. This is to be expected as authors use specific words only after they believe the words were acquired by a large portion of the intended audience.

3.2 Comparison with Oral Studies

Among the related work in the linguistic community, are several studies concerning oral acquisitions of words. These studies estimate the age at which a word is acquired for oral use based an interview processes with participating adults. We focus specifically on the seminal study of acquisition ages performed by Gilhooly and Logie (GL) (1980) and made available through the MRC database (Coltheart, 1981).

There are some substantial differences between these previous studies and our approach. We analyze the age acquisition process through document readability which leads to a written, rather than oral, notion of word acquisition. Furthermore, our estimates are based on documents written with a specific audience in mind, while the previous studies are based on interviewing adults regarding their childhood word acquisition process which is arguably less reliable due to the age difference between the acquisition and the interview. Finally, the GL study was performed in the late 1970s while our study uses contemporary internet data. Conceivably, the word acquisition process changed over the past 3 decades.

Despite these differences, it is interesting to contrast our inferred age acquisitions with the GL study and consider the differences and similarities. Figure 3 displays the relationship between the GL age of acquisition (AoA) and the acquisition ages obtained from readability data based on the $s = 0.8$ quantile. Some correlation is present ($r^2 = 0.34$) but the two measures differ considerably. As expected, the acquisition ages obtained from written readability data tend to be higher than the oral studies. The distributions of differences between the GL acquisition ages and the ones inferred from the readability data appears in Figure 4.

Comparing the acquisition ages obtained from readability data to the GL study results in a mean absolute error of 0.9 to 1.5, depending on the spe-

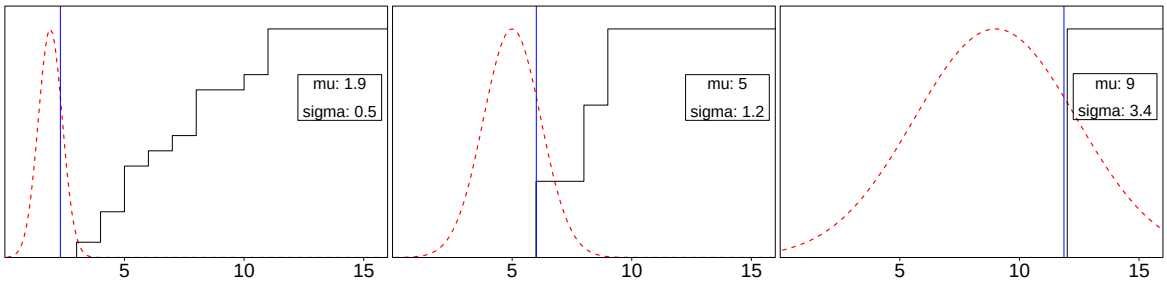


Figure 2: A comparison of empirical word appearances and AoA distributions for three words: `thought` (left), `multitude` (middle), and `assimilation` (right). The empirical cdf of word appearances appears as a piecewise constant line and the estimated pdf is indicated by the dashed curve with its 0.8 quantile indicated by a vertical line.

cific value of the Rasch parameter θ. Interestingly, the tendency for the written acquisition age to exceed the oral one diminishes as the grade level increases. This represents the notion that at higher grades words are acquired in both oral and written senses at the same age.

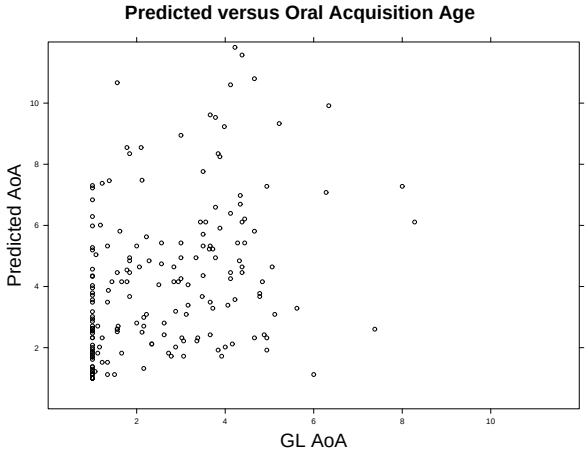


Figure 3: A scatter plot ($s = 80$, $n = 50$) of predicted age of acquisition versus Gilhooly and Logie's values reveals the tendency for the written estimate to exceed the oral estimate ($r^2 = 0.34$).

A comparison to two more recent studies confirms relationships that are similar to those observed with GL AoA. The Bristol Norm study (Stadthagen-Gonzalez and Davis, 2006) was performed in an identical way to the GL study and comparing the lists of acquisition ages results in a mean absolute error of approximately 0.5 which is much lower than the .9 to 1.5 relative to GL. The recent AoA list of Cortese and Khanna (2008) showed an increase in correlation relative to the GL study ($r^2 = 0.43$) potentially reflecting change in the acquisition process due to temporal effects.

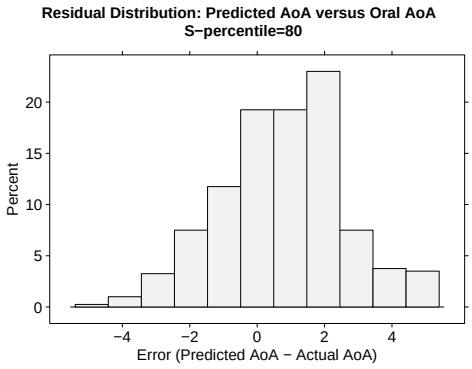


Figure 4: The difference distribution between the GL and the inferred AoA from Web 1-12 is skewed to the right as would be expected since written AoA is higher than oral AoA. Relaxing the definition of readability by decreasing s results in higher inferred acquisition ages. Values of s in $[0.5, 0.9]$ produced reasonable results, with $s = 0.65$ achieving smallest mean absolute error.

Those words that have the same written and verbal acquisition age are partially attributable to those words learned prior to first grade. Many words are learned between the ages of 2 and 5, while reading materials are typically not assigned a grade level of less than 1 or age 6. Approximately 40% of the words assigned the same grade level by both Gilhooly and our prediction had an AoA of 1st grade.

In some cases, the ages of acquisition obtained from readability data is actually lower than the ages reported in the older oral studies. This phenomenon is likely caused by a combination of a shift in educational standards, a change in social standards, or estimation errors due to sample size and modeling assumptions. Approximately

30 years have passed since Gilhooly and Logie's study was conducted. Specifically, society has made efforts to enhance the safety and health of children and to increase the attention to science education in very early grades. For example, the word `drug` appeared in writing 0.94 grades earlier than the age in which it was acquired orally according to the GL study. The newer Bristol Norm study confirms this observation as it predicts a decrease in grade level for `drug` of 0.88 over GL as well. A similar decrease in acquisition age relative to the GL norms was noted for many other words such as `hypothesis`, `conclusion`, `engineer`, `diet`, `exercise`, and `vitamin`.

3.3 Global Readability Prediction

Once acquisition age distributions are available, whether estimated statistically from data or obtained from a survey, they may be used to predict the grade level of novel documents. Specifically, the model predicts readability level t^* for a novel document d if it is the minimal grade for which readability is established:

$$t^* = \min\{t : P(d \text{ is readable at age } t) \geq \beta(t)\} \tag{5}$$

where $\beta(t)$ is a parameter describing the strictness of the readability requirement. Note that we allow $\beta(t)$ to vary as a function of time (grade level). We discuss the justification for this below.

A critical issue for reading difficulty prediction is how to handle words that appear in a new document that have never been seen in the training/development texts. In a statistical approach, the solution to this smoothing problem has two steps. First, we must decide how much total probability mass to allocate to all unknown words. Second, we must decide how to subdivide this total mass for individual words or classes of words using word-specific priors.

Our experience suggests that the first step of estimating total probability mass is particularly important: the likelihood of seeing an unknown word increases as a function of total vocabulary size, which is continuously growing with time. We model this by defining the following dynamic threshold

$$\beta(t) = \frac{\exp(at - 0.5)}{1 + \exp(at - 0.5)}. \tag{6}$$

We learn the growth rate parameter a in (6) from the data at the same time as we learn the readability model's quantile parameters $s = 1 - \epsilon_1$, $r = 1 - \epsilon_2$. The range of the resulting $\beta(t)$ is typically 0.5 in lower grades, increasing to 0.9 in higher grades. We discuss fitting these parameters and their optimal values further in Sec. 3.3.1. We found that using any fixed β value for all grades was generally much less effective than a dynamic $\beta(t)$ threshold, and so we focus on the latter in our evaluation.

For the second (word-specific) smoothing step, we simply assign uniform probability across grades, once the total unseen mass is determined. More sophisticated word-specific priors incorporating word length, morphological features, semantic clusters and so on are certainly possible and an interesting direction for future work.

In the following section we conduct three experiments involving readability prediction. First, we confirm the effectiveness of the AoA-based model compared to other predictive models. Second, we examine how prediction effectiveness is affected when our learned (written) acquisition ages are replaced with existing oral AoA norms. Third, we examine the ability of our model to generalize to new content by training and testing on different (non-overlapping) corpora.

3.3.1 Effectiveness of Readability Prediction

In order to assess the effectiveness of our model in predicting the readability grade levels of novel documents we apply the model to two corpora. First, we use the Web 1-12 corpus to learn optimal parameter values for a , r, and s and then assess prediction error using a test-training paradigm for the proposed model, Naive Bayes, and support vector regression. Second, the trained model is applied with to the Reader A-Z corpus and the results are compared with alternative semantic variables. Because corpora can vary significantly in text homogeneity, amount of noise, document size, and other factors, training and testing across different corpora – rather than relying on cross-validation with a single pooled dataset – gives valuable information about how a prediction method might be expected to perform on data with widely different characteristics. This particular choice of Web 1-12 for training and ReadingA-Z for testing was arbitrary.

To evaluate the best values for the a parameter in (6) and s, r parameters in Definition 1 we gen-

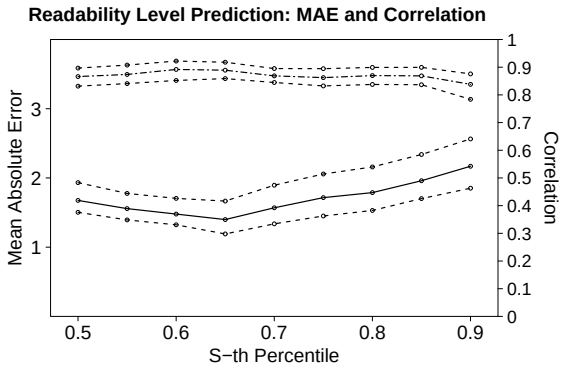


Figure 5: Mean absolute error (MAE) and correlation coefficient as functions of the quantile parameter s at optimal levels of a and r, averaged over 100 training/test samples. The MAE is displayed as the solid line and is aligned with the left axis while the correlation is displayed as a dashed line and is aligned with the right axis. 90% bootstrap confidence intervals are displayed.

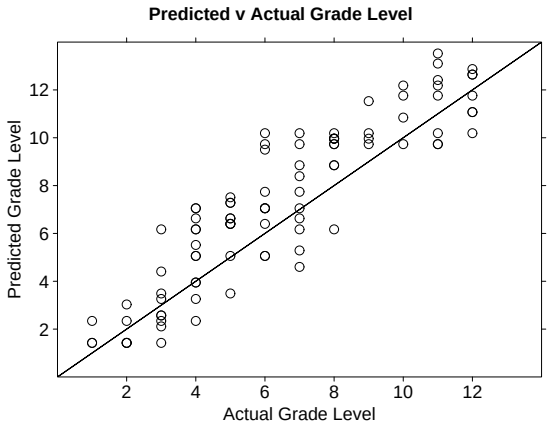


Figure 6: The scatter plot demonstrates the strong relationship between predicted and actual global readability levels.

Prediction Rule	MAE	LB	UB
Age of Acquisition	1.40	1.19	1.67
Naive Bayes	1.98	1.71	2.26
SVR (word frequency)	1.86	1.69	2.06
SVR (AoA percentiles)	1.36	1.22	1.58
Grade 6	2.92	-	-

Figure 7: A comparison of mean absolute error (MAE) across prediction algorithms shows the age of acquisition model compares favorably. The confidence bounds (LB,UB) were computed by repeating each model building procedure 100 times.

erated 100 independent test and training samples and computed the mean absolute prediction error (MAE) and the correlation coefficient between the predicted and actual levels. Figure 5 (left) shows these two quantities: in each group of three lines, the top and bottom lines delineate the upper and lower 90% confidence bounds for the middle line. Each middle line gives mean error or correlation as a function of the quantile parameter s at optimal levels of r and a, averaged over the 100 training/test samples. The optimal value of s for both quantities is around 0.6 (0.65 for the MAE). The optimal value for parameter a was approximately 1.55. The best MAE is 1.4 which compares favorably to the 2.92 MAE obtained by always predicting Grade 6 which is the optimal "dumb" classifier in the sense that of all constant predictors it provides the smallest expected MSE over a uniform grade distribution as is the case with the Web1-12 corpus. Figure 6 is a scatter plot comparing predicted grades vs. actual grades, with a strong correlation of 0.89.

We compared the predictions of model (3) to two standard classifiers: naive Bayes and support vector regression (SVR). SVR was applied twice using different sets of features - once with the document word frequencies and once with the estimated AoA percentiles for the document words. The document word frequency vector is comparable to the semantic component of the machine learning approach used by (Heilman et al., 2008). The 75-25 training-test model building paradigm was used over documents from grades 1 to 12 to obtain predicted values. The MAE for these predictors and their 90% confidence intervals are shown in Figure 7. Predicting readability using word frequencies had inferior performance, with the naive Bayes model performing poorly and the SVR and Rasch model obtaining MAE around 1.4.

In the second experiment, we compared our model to published correlation results (Collins-Thompson and Callan, 2005) for multiple alternative semantic variables using the same Reading A-Z corpus, with the results shown in Fig. 8. Details on these semantic variables, which have been used in previous statistical learning approaches, are available in the same study. Interestingly, the correlation of the model was comparable to ex-

	Correlation		Correlation
GL (Web)	.65	UNK	.78
GL (WR)	.40	Type	.86
Bristol (Web)	.76	MLF	.49
Bristol (WR)	.57	FK	.30
Inferred (Web)	.59	Unigram	.63

Figure 8: Comparison of the correlation of AoA and other semantic variables with grade level for the Reading A-Z corpus, showing the AoA model with the dynamic threshold compares well to existing methods. The competitor methods used are from (Collins-Thompson and Callan, 2005) and comprise the Smoothed Unigram, UNK (relative to revised Dale-Chall), TYPE (number of unique words), MLF (mean log frequency), and FK (Flesch-Kincaid readability).

isting variables, but did vary depending upon the source of AoA. Note that because the Reading A-Z texts were assigned grades by their creators using some of the same semantic variables (e.g. Type), it is not surprising that those variables perform especially well on this dataset.

High quality readability prediction is a worthwhile result in itself; however, we can also use the prediction mechanism to study the validity of Definition 1 and the Rasch model. We do so by applying other predictive algorithms using the inferred acquisition age distribution for each document as the predictor variables and comparing the MAE with the MAE obtained by the estimated Rasch model. In particular, we examine the performance of support vector regression (SVR) using the estimated AoA percentiles for each document as predictor variables. The results displayed in Figure 7 show that SVR and the dynamic threshold prediction rule perform similarly well, suggesting that Definition 1 and the Rasch model are suitable models for readability prediction.

3.3.2 Prediction with Existing Acquisition Age Norms

We now examine how predicting readability of novel documents using acquisition ages obtained in surveys perform in comparison to the ages obtained from the maximum likelihood estimation.

We use the GL and Bristol age of acquisition norms. The intersection of AoA norm data and the Web Corpus are 1217 and 1012 words respectively for the GL and Bristol measure; additionally, the highest grade level associated with these word sets

Prediction Rule	S-th Percentile	Dynamic Threshold
Age of Acquisition	1.69	1.40
GL Norms	1.73	1.42
Bristol Norms	1.97	1.79

Figure 9: The Gilhooly and Logie AoA norms and the Bristol norms are independent sources for ages of acquisition. A comparison of the prediction quality using these norms shows two things: 1) the definition provides comparable prediction quality using expert norms, and 2) the dynamic threshold $\beta(t)$ improves prediction over the static threshold (optimal s-th percentile) for the norms.

AoA Source	Web 1-12	Weekly Reader
Inferred (Weekly Reader)	-	.91
Inferred (Web 1-12)	1.89	-
GL	2.05	1.14
Bristol	1.57	1.34

Figure 10: The readability of WR documents was predicted using 4 sources of AoA data. The parameters of the prediction model were fit using only the Web data, or the WR data, or both sources in the case of the GL and Bristol norms AoA data.

are eight and seven respectively. When applying the prediction rule using AoA norms r is implicitly selected in the norming process as the result is a single value instead of a distribution. Interestingly, the optimal ranges of s-percentile, from 92 to 100, were the same for both the GL and Bristol norms. Table 9 shows that the prediction accuracy obtained using the GL Norms was almost identical to that obtained with the inferred AoA, while the Bristol Norms performed as well as some of the competitor procedures.

3.3.3 Prediction Effectiveness across Different Corpora

To provide additional evidence for our model's ability to generalize to new corpora, we examine how the learned r and s values vary when the model is learned on one corpus and evaluated on another, and how this affects the accuracy of the readability prediction.

Figure 10 demonstrates the corpus used for tuning the readability prediction has a large impact on the quality of the prediction. Comparing the MAE of the readability predictions on WR data

when the age of acquisition is inferred from Web data to the MAE when the AoA is inferred from WR data shows the error rate more than doubles from 0.90 to 1.89. The increase in error rate also appears when the age of acquisition for WR data is predicted using the AoA norm data. In this case the prediction was performed using the parameters identified when the model was trained on Web data and when the model was trained on WR data. In each case a tendency to overfit appears as the MAE increases from 1.14 to 2.05 for the GL norms and 1.34 to 1.57 for the Bristol norms. Interestingly, the Bristol norms perform better on WR data when fit using the Web data, while the GL norms perform better when fit using the WR data.

4 Related Work

Age of acquisition for word reading and understanding has been extensively studied as a learning factor in the psycholinguistics literature, where AoA norms have been obtained using surveys. Examples of relevant literature are (Gilhooly and Logie, 1980; Zevin and Seidenberg, 2002). Our approach differs by connecting AoA to readability through Definition 1 and using readability data to estimate AoA norms from large amounts of authentic language data. A related study is that by Crossley et al. (2007) who used AoA to help discriminate between authentic and simplified texts for second-language readers.

In the past decade, there has been renewed interest in corpus-based statistical models for readability prediction. One example is the popular Lexile measure (Stenner, 1996) which uses word frequency statistics from a large English corpus. Collins-Thompson and Callan (2005) introduced a new approach based on statistical language modeling, treating a document as a mixture of language models for individual grades. Further recent refinements in methods for readability prediction include using machine learning methods such as Support Vector Machines (Schwarm and Ostendorf, 2005), log-linear models (Heilman et al., 2008), k-NN classifiers and combining semantic and grammatical features (Heilman et al., 2007).

The growing number of features investigated by these machine learning approaches reflect the fact that reading difficulty is a complex phenomenon involving many factors, from semantic difficulty (vocabulary) to syntax and discourse complexity, reader background, and others. While a full-featured comparison between previous approaches that includes AoA features would be very interesting, our goal in this study was to provide a clear analysis of the most fundamental factor of readability, semantic difficulty, which accounts for 80-90% of the variance in readability prediction scores (Chall and Dale, 1995). Because AoA is a semantic, vocabulary-based representation, we compare its effectiveness with the corresponding *semantic components* from previous machine-learning approaches in Sec. 3.3.1.

5 Discussion

While there have been several recent studies regarding word acquisition and readability our work is the first to provide a quantitative connection between these two concepts in a statistically meaningful way. The core assumption that we make is Definition 1 which is consistent with standard readability definitions e.g., (Chall and Dale, 1995) and states that document readability level is determined by most people understanding most words.

The connection between word acquisition and readability is both intuitive and useful. It allows two degrees of freedom $s = 1 - \epsilon_1$ and $r = 1 - \epsilon_2$ to handle situations where different readability notions exist. Experiments validate the model and demonstrate interesting trends in word acquisitions as compared to older oral acquisition studies. Experimental results show that the proposed model is also effective in terms of predicting readability level of documents on multiple datasets. It compares favorably to naive Bayes and support vector regression, the latter being one of the strongest regression baselines.

Acknowledgments

The authors thank Joshua Dillon for downloading the weekly reader data and pre-processing it. The work described in this paper was funded in part by NSF grant DMS-0604486.

References

J. S. Chall and E. Dale. 1995. *Readability Revisited: The New Dale-Chall Readability Formula*. Brookline Books, Brookline, MA.

K. Collins-Thompson and J. Callan. 2005. Predicting reading difficulty with statistical language models. *J. of the American Soc. for Info. Science and Tech.*, 56(13):598–605.

M. Coltheart. 1981. The MRC psycholinguistic database. *Quarterly Journal of Experimental Psychology*, 33A:497–505.

M. Cortese and M. Khanna. 2008. Age acquisition ratings for 3000 monosyllabic words. *Behavior Research Methods*, 40:791–794.

S. A. Crossley, P. M. McCarthy, and D. S. McNamara. 2007. Discriminating between second language learning text-types. In *Proc. of the Twentieth International Florida Artificial Intelligence Research Society Conference*.

H. A. David and H. N. Nagaraja. 2003. *Order Statistics*. Wiley, Marblehead, MA.

E. Fry. 1990. A readability formula for short passages. *Journal of Reading*.

K. J. Gilhooly and R. H. Logie. 1980. Age of acquisition, imagery, concreteness, familiarity and ambiguity measures for 1944 words. *Behaviour Research Methods and Instrumentation*, 12:395–427.

M. Heilman, K. Collins-Thompson, J. Callan, and M. Eskenazi. 2007. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Proc. of the Human Language Technology Conference*.

M. Heilman, K. Collins-Thompson, and M. Eskenazi. 2008. An analysis of statistical models and features for reading difficulty prediction. In *The 3rd Workshop on Innovative Use of NLP for Building Educational Applications*.

S. E. Schwarm and M. Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proc. of the Association of Computational Linguistics*.

H. Stadthagen-Gonzalez and C. J. Davis. 2006. The bristol norms for age of acquisition, imageability, and familiarity. *Behavior Research Methods*, 38:598–605.

A. J. Stenner. 1996. *Measuring reading comprehension with the Lexile Framework*. Metametrics, Inc., Durham, NC.

J. D. Zevin and M. S. Seidenberg. 2002. Age of acquisition effects in word reading and other tasks. *Journal of Memory and Language*, 47(1):1–29.

Combining Collocations, Lexical and Encyclopedic Knowledge for Metonymy Resolution

Vivi Nastase and **Michael Strube**
EML Research gGmbH
Heidelberg, Germany
http://www.eml-research.de/nlp

Abstract

This paper presents a supervised method for resolving metonymies. We enhance a commonly used feature set with features extracted based on collocation information from corpora, generalized using lexical and encyclopedic knowledge to determine the preferred sense of the potentially metonymic word using methods from unsupervised word sense disambiguation. The methodology developed addresses one issue related to metonymy resolution – the influence of local context. The method developed is applied to the metonymy resolution task from SemEval 2007. The results obtained, higher for the countries subtask, on a par for the companies subtask – compared to participating systems – confirm that lexical, encyclopedic and collocation information can be successfully combined for metonymy resolution.

1 Introduction

Metonymies are a pervasive phenomenon in language. They occur because in communicating, we use words as pointers to a larger body of knowledge, that encompasses various facets of the concept evoked by a given word.

A listener need not understand the cello to be moved by its playing, just as it is unnecessary for a rider to understand technical jargon; all that matters is sensation, and here the **Kawasaki** *excels. The* **cockpit** *is sensibly designed, with a narrow* **front seat** *portion ...*

Kawasaki is a company, it has an organization, facilities, employees, it makes specific products. In the context above, the company name stands in for its products – motorcycles. Motorcycles have parts, the *cockpit* and *front seat* are some of them, and this provides the discourse links between the two sentences. Constraints on the interpretation of a word w in context comes both from the local and global context, and are applied to the information/knowledge evoked by w. The local constraints come from the words with which w is (grammatically) related to. The global constraints come from the domain/topic of the text, discourse relations that span across sentences.

Metonymic words have a rather small number of possible interpretations (also called readings) which occur frequently (Markert and Nissim, 2002). Idiosyncratic interpretations are also possible, but very rare. One can view the possible interpretations of a potentially metonymic word (PMW) as corresponding to the word's possible senses (Nissim and Markert, 2003), bringing the task close to word sense disambiguation.

The approach to metonymy resolution presented here is supervised, with unsupervised feature enrichment. We apply techniques inspired by unsupervised word sense disambiguation, which allow us to go beyond the annotated data provided in training, and quantify the restrictions imposed on the interpretation of a PMW by its grammatically related neighbours through collocation information extracted from corpora. The only annotation required for the corpora are automatically induced part-of-speech tags from which we obtain grammatical relations through regular expression matching over sequences of parts-of-speech. Collocation information is combined with lexical resources – WordNet – and encyclopedic knowledge extracted from Wikipedia to help us generalize the collocations found to determine higher level constraints on a word's grammatical collocates. In the example above, *Kawasaki* is grammatically related to the verb *excel* – it is its subject. To determine the most likely interpretation of *Kawasaki* given that it is in the subject relation with *excel* we look at all the nouns in the corpora

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 910–918,
Singapore, 6-7 August 2009.

that appear as this verb's subjects, and estimate from this the preferences *excel* has for its subjects. Let us say the corpus contains the following collocations in subject position (with frequency information in parentheses): *player (4), musician (50), car (30), computer (12), camera (40), driver (55), bike (20)* .... The knowledge resources – WordNet, *isa* relations extracted from Wikipedia – will help generalize these collocations: *player, musician, driver* to `person` and *car, computer, camera, bike* to `artifact`. This together with frequency of occurrence are used to estimate the probability that the verb *excel* takes a `person` or `artifact`-type subject. These are *excel*'s selectional preferences towards certain collocates, and will help determine which possible interpretation for the PMW *Kawasaki* is appropriate in this context – `organization-for-people` or `organization-for-product`.

The paper continues with related work in Section 2 and the description of the data in Section 3. The representation used is introduced in Section 4. The results and the discussion are presented in Section 5. The paper wraps up with conclusions and future work.

2 Related Work

Analysis of metonymies as a linguistic phenomenon dates back at least to the 1930s (Stern, 1931), and are increasingly recognized as an important phenomenon to tackle in the interest of higher level language processing tasks, such as anaphora resolution (Harabagiu, 1998; Markert and Hahn, 2002), question answering (Stallard, 1993) or machine translation (Kamei and Wakao, 1992).

Until the early 90s, the main view about metonymies was that they violate semantic constraints in their immediate context. To resolve metonymies then amounts to detecting violated constraints, usually from those imposed by the verbs on their arguments (Pustejovsky, 1991; Hobbs et al., 1993; Fass, 1991). Markert and Hahn (2002) showed that this approach misses metonymies which do not violate selectional restrictions. In this case referential cohesion relations may indicate that the literal reading is not appropriate and give clues about the intended metonymic interpretation.

Markert and Nissim (2003) have combined observations from the linguistic analysis of metonymies with results of corpus studies. Linguistic research has postulated that (i) conventional metonymic readings are very systematic; (ii) unconventional metonymies can be created on the fly and their interpretation is context dependent; (iii) metonymies are frequent. The fact that most metonymic interpretations are systematic and correspond to a small set of possible readings allow the metonymy resolution to be modelled as a classifier learning task. Markert and Nissim (2002) and Nissim and Markert (2003) have shown that conventional metonymies can be effectively resolved using a supervised machine learning approach. Moreover, grammatically related words are crucial in determining the interpretation of a PMW. The shortcoming is that manually annotated data is in short supply, and the approach suffers from data sparseness. To address this problem, Nissim and Markert (2003) proposed a word similarity-based method. They use Lin's thesaurus (Lin, 1998) to determine how close two lexical heads are, and use this instead of the more restrictive identity constraint when comparing two instances. This technique is complex, requiring smoothing, multiple iterations over the thesaurus and hybrid methods to allow a back-off to grammatical roles.

The supervised approach to resolving metonymies was encouraged by the metonymy resolution task at the semantic evaluation exercise SemEval 2007 (Markert and Nissim, 2007). The participating systems in this task were varied. Most of them (four out of five) have used supervised machine learning techniques. The systems that beat the baseline used either the grammatical annotations provided by the organizers (Farkas et al., 2007; Nicolae et al., 2007), or a robust and deep (not freely available) parser (Brun et al., 2007). These systems represented instances in a manner similar to (Nissim and Markert, 2005). They used additional manually built resources – WordNet, FrameNet, Levin's verb classes, manually built lists of "trigger" words – to generalize the existing features. Brun et al. (2007) also used the British National Corpus (BNC) for computing the distance between words based on their syntactic distribution.

While lexical resources and corpora are used to estimate word similarity, all these systems rely exclusively on the data provided by the organizers – instance representation captures only information that can be derived from or between the data points provided. The approach presented here goes beyond the given data, and induces from corpora measures that allow the system to determine

what are the preferences of the words surrounding a PMW towards each of PMW's possible readings. The technique employed is adapted from unsupervised word sense disambiguation (WSD). In short, we use the local grammatical context as it is commonly used in WSD approaches, to guide the system in choosing the reading that fits best. The benefits of using grammatical information for automatic WSD were first explored by Yarowsky (1995) and Resnik (1996) in unsupervised approaches to disambiguating single words in context. The method described here uses automatically induced selectional preferences, computed from sense-untagged data, similar to Nastase (2008).

3 Data

We work with the data from the metonymy resolution task at SemEval 2007 (Markert and Nissim, 2007), generated based on a scheme developed by Markert and Nissim (2003).

The metonymy resolution task at SemEval 2007 consisted of two subtasks – one for resolving country names, the other for companies. For each subtask there is a training and a test portion. Figure 1 shows the text fragment for one sample, and Table 1 the data statistics. The *reading* column shows the possible interpretations of a PMW for countries and companies respectively. For example, `org-for-product` would be the interpretation of the PMW *Kawasaki* in the example shown in the introduction.

Occurrences of country and company names were annotated with a small number of possible readings, as shown in Table 1. This reflects previous analyses of the metonymy phenomenon, which showed that there is a rather small number of possible interpretations that appear more frequently (Markert and Nissim, 2002). Special interpretations are very rarely encountered.

Within the framework of the SemEval task, metonymy resolution is evaluated on the given test data, on three levels of granularity: coarse – distinguish between `literal` and `non-literal` readings; medium – distinguish between `literal`, `mixed` and `non-literal` readings; fine – identify the specific reading of the target word/words (potentially metonymic word - PMW).

reading	train	test
locations	925	908
literal	737	721
mixed	15	20
othermet	9	11
obj-for-name	0	4
obj-for-representation	0	0
place-for-people	161	141
place-for-event	3	10
place-for-product	0	1
organizations	1090	842
literal	690	520
mixed	59	60
othermet	14	8
obj-for-name	8	6
obj-for-representation	1	0
org-for-members	220	161
org-for-event	2	1
org-for-product	74	67
org-for-facility	15	16
org-for-index	7	3

Table 1: Reading distributions

4 Representation

The method presented in this paper is a supervised learning method, along the same general lines as the systems which participated in the SemEval competition. As such, it represents each PMW in the data through features that describe its context and some semantic characteristics. The minimum set of necessary features is taken to be that presented by Nissim and Markert (2005), and proved to be effective in solving metonymies. These are the *M&N features* (Markert and Nissim features). We expand on these features and estimate preferences from words in a PMW's context towards specific PMW interpretations. These constitute the *selectional preference features*. Finally, Wikipedia is a source of facts which can be used to derive information that can bias the decision towards certain interpretations for a PMW. Each of these features are described in more detail in the following subsections.

4.1 M&N features

The features used by Nissim and Markert (2005) are:

- grammatical role of PMW (subj, obj, ...);
- lemmatized head/modifier of PMW (announce, say, ...);
- determiner of PMW (def, indef, bare, demonst, other, ...);

XML tagged text

<sample id="samp114">
<bnc:title> Computergram international </bnc:title>
<par>
LITTLE FEAR OF MICHELANGELO
The computer industry equivalent of "Small earthquake in Chile" ...
The Michelangelo computer virus that received worldwide attention last year is expected to cause even fewer problems this Saturday than it did when it struck last year, a team of <annot><org reading="literal"> IBM </org></annot> researchers said.
</par>
</sample>

Grammatical annotations

SampleID|Lemma|PMW|GrRole|Reading
samp114|researcher|IBM|premod|literal
samp4|be|Williams Holdings|subj|literal
samp5|parent|Fujitsu Ltd|app|mixed
samp5|have|Fujitsu Ltd|subj|mixed
samp5|keep|Fujitsu Ltd|subj|mixed
samp8|against|IBM|pp|literal

POS tags

<bnc:s id="samp114-bncCNJ-s341"> ...
<bnc:w id="samp114-bncCNJ-s343-w29" bnc:type="NN0"> team </bnc:w> <bnc:w id="samp114-bncCNJ-s343-w30" bnc:type="PRF"> of </bnc:w> <annot> <org reading="literal"> <bnc:w possmeto="yes" id="samp114-bncCNJ-s343-w31" bnc:type="NP0"> IBM </bnc:w> </org> </annot> <bnc:w id="samp114-bncCNJ-s343-w32" bnc:type="NN2"> researchers </bnc:w> ...

Figure 1: Sample annotation

- grammatical number of PMW (sg, pl);
- number of grammatical roles in which the PMW appears in its current context;
- number of words in PMW;

All these features can be extracted from the grammatically annotated and POS tagged data provided by the organizers.

4.2 Selectional preference features

The grammatical relations and the connected words are important to describe the local context of the target PMW. Because of the limited amount of annotated data (a few thousand instances), lemmas of PMW's grammatically related words will make for very sparse data that a machine learning system would not be able to generalize over. Nissim and Markert (2003) and the teams participating in the metonymy resolution task have then supplemented their systems with Lin's thesaurus, WordNet, Beth Levin's verb groups, FrameNet information, or manually designed lists of words to generalize the grammatically related words and thus find shared characteristics across instances of metonymies in text.

The notion of selectional restrictions used in metonymy resolution – meaning the restrictions imposed on the interpretation of a PMW by its context – is similar to the notion of selectional preferences from word sense disambiguation – meaning the preferences of a word for the senses of the words in its context. We import this notion, and compute selectional preferences for the words in a PMW's (grammatical) neighbourhood, and allow them to influence the chosen reading for the PMW. Applying methods from unsupervised WSD allow us to estimate such preferences from (sense/metonymy) untagged corpora.

A potentially metonymic word (or phrase) has a small number of possible readings. These can be viewed as possible senses, and the task is to choose the one that fits best in the given context. The preference for each possible sense can be determined based on the PMW's grammatically related words. To estimate these sense preferences we use grammatical collocations extracted from the British National Corpus (BNC), detected using regular expression matching over sequences of POS using the Word Sketch Engine (Kilgarriff et al., 2004). The scores are computed following a technique similar to Nastase (2008), which is illustrated using the following example:

The Kawasaki drives well, steers brilliantly both under power and in tight corners ...

The PMW *Kawasaki* is involved in the following grammatical relations in the previous sentence:

(drive,subject,Kawasaki)
(steer,subject,Kawasaki)

SampleID	Lemma	PMW	GrRole	Reading	act	animal	artifact	...	person	...
samp190	say	Sun	subj	org-for-members	0.00056	0.01171	0.01958	...	0.61422	...
samp190	claim	Sun	subj	org-for-members	0.00198	0.00099	0.00893	...	0.50211	...

Table 2: Grammatical annotation file enhanced with selectional preference estimates

The BNC provides the collocations *(drive,subject,X)* and *(steer,subject,Y)*, to determine what kind of subject *drive* and *steer* prefer, in "word-POS:frequency" format:

drive subject chauffeur-n:12, engine-n:30, car-n:62, taxi-n:13, motorist-n:10, disk-n:15, truck-n:11, man-n:75, ...

steer subject power-n:6, car-n:3, sport-n:2, firm-n:2, boy-n:2, government-n:2, man-n:2, people-n:2 ...

The target whose interpretation must be determined is *Kawasaki*. If for a potentially metonymic word representing a company name, there are the following possible interpretations: `company`, `member/person`, `product/artifact`, `facility`, `name`, we compute the preference for each of these interpretations based on the extracted collocations. For the verb *drive* for example, the collocations *engine, car, taxi, truck* are all *artifacts* (according to WordNet), and thus vote for the `product/artifact` reading, while *chauffeur, motorist, man* are all *person*, and vote for the `member/person` reading. Preferences from different grammatical relation for the same PMW are summed.

Formally, we choose the PMWs' "senses" – a set of words which are close to the possible readings of metonymic words in the data. In this work, these senses are the WordNet 3.0 supersenses:

`S = { act, animal, artifact, attribute, body, cognition, communication, event, feeling, food, group, location, motive, object, person, phenomenon, plant, possession, process, quantity, relation, shape, state, substance, time }`.

Because none of these can be seen as a sense for "company", the list is supplemented with `company` and `organization`. Granted, there is no 1:1 mapping from these supersenses to PMW readings, but find such a strict correspondence is not necessary because the context preferences for each of these senses are used as features, and the mapping to PMW readings is found through a supervised learned model.

To compute the preference of a word w in the grammatical context of a PMW t (the target) towards each of t's possible senses, we consider each relation (w, R, t), where R is the grammatical relation. The set C of word collocations are extracted from the BNC

$$C = \{(w, R, w_j : f_j) | (w, R, w_j) \in BNC, \\ f_j \text{is the frequency of occurrence}\}$$

and used to compute a preference score P_{s_i} for each sense $s_i \in \mathsf{S}$:

$$P_{s_i} = \frac{\sum_{(w,R,w_{i,j}:f_{i,j}) \in C_{s_i}} f_{i,j}}{\sum_{(w,R,w_j:f_j) \in C} f_j}$$

where

$$C_{s_i} = \{(w, R, w_j : f_j) | (w, R, w_j : f_j) \in C; \\ \text{supersense}(w_j, s_i) \parallel \text{isa}(w_j, s_i)\}.$$

supersense(w_j, s_i) is true if s_i is a supersense of one of w_j's senses;

isa(w_j, s_i) is true if s_i is a hypernym of one of w_j's senses in WordNet, or is a fact extracted from Wikipedia.

To determine the *supersense* and *isa* relation we use WordNet 3.0, and a set of 7,578,112 *isa* relations extracted by processing the page and category network of Wikipedia[1] (Nastase and Strube, 2008). The collocations extracted from BNC contain numerous named entities, most of which are not part of WordNet. If an *isa* relation between a collocate from the corpus w_j and a possible sense of a PMW s_i cannot be established using supersense information (for the supersenses) or through transitive closure in the hypernym-hyponym hierarchy in WordNet (for `company`

[1] `http://www/eml-research.de/nlp/download/wikirelations.php`

and `organization`) for any sense of w_j, it is tried against the Wikipedia-based links.

This process transforms the grammatical annotation file and enhances it with the collocation estimates, as shown in Table 2 (compare this with a sample of the original file presented in Figure 1).

4.3 Product and event features

Farkas et al. (2007) observed that using the PMWs themselves as features leads to improvement on determining the reading for organization names, and postulate that this is because some company names are more likely to be used in a metonymic way. This is often the case with companies that make products which are commonly used (cars, for example).

Brun et al. (2007) note that certain locations, such as *Vietnam*, are more likely to be used with an *event* reading than others locations. Generally, locations strongly associated with events tend to be used to refer to the event, and more often have a `place-for-event` interpretation rather than a `literal` one.

These two observations have lead us to mine for these pieces of information in the Wikipedia relations, and to add two more features for a target PMW:

has-product will take a value of 1 if any of the PMW's hypernyms (according to the *isa* relations extracted from Wikipedia) contains the string *manufacturer*, will have the value 0 otherwise;

has-event will have the value 1 if any of the PMW's hypernyms refers to an event (movements/operations/riots), and value 0 otherwise.

4.4 Data representation

As mentioned before, the representation built can be seen as consisting of roughly three subsets of features:

- **the M&N features** proposed by Nissim and Markert (2005). To combine the grammatical information from all relations, we transform the grammatical relations into features (as opposed to values). For a relation *subject* for example, we generate a binary `subject` feature that indicates whether for a given target this grammatical relation is filled or not, and a `subject_lemma` feature , whose value is the lemma of the grammatically related word.

- **the selectional preference scores**. Each of these features corresponds to one of the elements of S, presented above. These features combine the selectional preferences of all the grammatical relations for one target PMW.

- **product and event information from Wikipedia** – has-product and has-event.

The grammatical annotation file consists of one entry for each grammatical relation in which a PMW appears. For the final representation, information about all relations of a given PMW is compressed into one instance. Because the basic features were binarized, and instead of having one *grammatical role* feature now each possible grammatical relation has its own feature, combining several entries for one PMW is easy, as it only implies setting the correct value for the grammatical relations that are valid in the PMWs context.

The final representation consists of 63 features + class feature for the subset for company PMWs, 59 features + class feature for the subset containing countries PMWs. The sample ID and the PMW itself were not part of this representation.

5 Results

The models for determining a PMW's correct interpretation are learned on the training data provided, and evaluated on the test portion, using the answer keys and evaluation script provided with the data. For learning the models we use Weka (Witten and Frank, 2005), and select the final learning algorithms based on 10-fold cross-validation on the training data. We have settled on support vector machines (SMO in Weka), and we use the learner's default settings.

Tables 3 and Table 4 show the results obtained, and the baseline and the best results from the SemEval task for comparison (Markert and Nissim, 2007). The baseline in Table 3 corresponds to classifying everything as the most frequent class – literal interpretation. The *M&N feat.* and *M&N feat.bin.* correspond to datasets that contain only the M&N features and the binarized versions of these features, respectively. *SemEval best* gives the best results obtained on each task in the SemEval 2007 task (Markert and Nissim, 2007). SMO_{wiki} are the results obtained with the complete feature set described in Section 4, and SMO_{SP} are the results obtained when only the new features are used – only selectional preference, has-product and has-event features (none of

task ↓ method →	baseline	SemEval best	SMO$_{wiki}$	SMO$_{SP}$	M&N$_{feat.}$	M&N$_{feat.bin.}$
LOCATION-COARSE	79.4	85.2	86.1	82.8	79.4	83.4
LOCATION-MEDIUM	79.4	84.8	85.9	82.6	79.4	82.3
LOCATION-FINE	79.4	84.1	85.0	82.0	79.4	81.3
ORGANIZATION-COARSE	61.8	76.7	74.9	66.6	73.8	74.0
ORGANIZATION-MEDIUM	61.8	73.3	72.4	65.0	69.8	69.4
ORGANIZATION-FINE	61.8	72.8	71.0	64.7	68.4	68.5

Table 3: Accuracy scores

task ↓ method →	base	max	SMO$_{wiki}$	SMO
LOCATION-COARSE				
literal	79.4	91.2	91.6	91.6
non-literal	20.6	57.6	59.1	58.8
LOCATION-MEDIUM				
literal	79.4	91.2	91.6	91.6
metonymic	18.4	58.0	61.5	61.5
mixed	2.2	8.3	16	8.7
LOCATION-FINE				
literal	79.4	91.2	91.6	91.6
place-for-people	15.5	58.9	61.7	61.7
place-for-event	1.1	16.7	0	0
place-for-product	1.1	0	0	0
obj-for-name	0.4	66.7	0	0
obj-for-rep	0	0	0	0
othermet	1.2	0	0	0
mixed	2.2	8.3	16	8.7
ORGANIZATION-COARSE				
literal	61.8	82.5	81.4	81.2
non-literal	38.2	65.2	61.6	60.7
ORGANIZATION-MEDIUM				
literal	61.8	82.5	81.4	81.2
metonymic	31.0	60.4	58.7	58.1
mixed	7.2	30.8	26.8	28.9
ORGANIZATION-FINE				
literal	61.8	82.6	81.4	81.2
org-for-members	19.1	63.0	59.7	59.2
org-for-event	0.1	0	0	0
org-for-product	8.0	50.0	44.4	44
org-for-facility	2.0	22.2	36.3	38.1
org-for-index	0.3	0	0	0
org-for-name	0.7	80.0	58.8	58.8
org-for-rep	0	0	0	0
othermet	1.0	0	0	0
mixed	7.2	34.3	27.1	29.3

Table 4: Detailed F-scores

the M&N features). The baseline for detailed reading results in Table 4 reflects the distribution of the classes in the test file. The *max* column shows the best performance for each task in the SemEval 2007 competition (Markert and Nissim, 2007). The SMO column shows the results of learning when Wikipedia information is not used to compute the values of the collocation, has-product and has-event features.

Nissim and Markert (2003) have shown that grammatical roles are very strong features. Experiments on the data represented exclusively through grammatical role features confirm this observation, as the results obtained using only the syntactic features (no lexical head information) give the same results as the *M&N feat.bin.* which does include lexical information.

On the location metonymies, the current approach performs better on all evaluation types (coarse, medium, fine) by 0.9, 1.1 and 0.9% points respectively. The improvement comes from recognizing better the metonymic readings, as it is apparent from the detailed F-score results in Table 4. For the coarse readings, the F-score for the `non-literal` reading is 1.5% points higher than the best performance at SemEval, and 2.5% and 7.7% points respectively for the `metonymic` and `mixed` readings for the medium and fine coarseness. It is interesting that the learning is quite successful even when only selectional preference and Wikipedia-based has-product and has-event features are used – the SMO_{SP} column in Table 3. The grammatical role and the related lemma were used to derive these collocation features, but they do not appear as such in the representation used for this batch of experiments.

For company metonymies the current approach does not perform better than the state-of-the-art. For these metonymies the syntactic information is not as useful. This is evidenced by the lower performance of the classifier that uses only syntactic information (column *M&N feat.bin.* in Table 3), despite the fact that the training dataset for com-

panies is larger than the one for countries. This observation is further supported by the low results when using only selectional preference features. It indicates that for company metonymies the local context does not provide as strong clues as it does for locations. For such PMWs we should explore the larger context. We have made a start with the Wikipedia-based features built following the observation about companies and their products made by Farkas et al. (2007) and Brun et al. (2007). In future work we plan to analyse this matter further, and find a method to derive more such features, and without manually provided clues (such as *manufacturer* or *riots*).

Wikipedia derived information does not contribute very much, but as expected it is helpful to identify other classes than the `literal` one. It is helpful to detect the `mixed` class – 16% F-score when using Wikipedia information compared to 8.7% for the countries data when we estimate preferences using only WordNet. It also increases the performance on the `non-literal`, `metonymic` and `org-for-members` classes in coarse, medium and fine classification respectively for both countries and companies. There is a small improvement for recognizing the `org-for-product` reading for organizations when using Wikipedia-based features. It is an indication that the has-product feature is useful. We cannot draw conclusions about the has-event feature, as there are only 3 training instances for the `place-for-event` reading. The results are encouraging, as we have just scraped the surface of the information that Wikipedia can provide.

The corpus derived selectional preferences perform very well, especially for determining the reading of locations. Analysis of the data and the features gives some indication as to why this happens: in the grammatical annotations provided, when the PMW is a prepositional complement or has a prepositional complement, the grammatically related word is a preposition. We extract only grammatical collocations for open-class words, restricted by the grammatical relation of interest, so we do not extract collocations for prepositions. Location prepositions (*in, at, from*) are less ambiguous than others (e.g. *for*), which are more common for the organization data. We have attempted to bypass this problem by generating parses using the dependency output of the Stanford Parser (de Marneffe et al., 2006), and bypassing the preposition – incorporate it in the grammatical role (*pp_in*, for example), and using as lemma the head of the prepositional complement or the constituent which dominates the prepositional phrase, depending on the position of the PMW. Now we can use the grammatical relation and the associated open-class word to look for collocations. This approach did not lead to good results, because the quality of the automatic parses is far from the manually provided information.

6 Conclusions

We have explored the use of selectional preference scores derived from a sense untagged corpus as local constrains for determining the interpretation of potentially metonymic words. Such methods were previously successfully used for word sense disambiguation, and transfer nicely to the metonymy resolution task. Adding encyclopedic knowledge to the mix improved the results further, by filling in gaps for WordNet, and extracting information particular to PMW. We plan to expand on this, and find methods to extract more such features automatically, without manually provided clues.

For a more comprehensive treatment of metonymies one must take into consideration not only local context but also discourse relations. A possible avenue of research is to build upon coreference resolution systems, and use the mentions they detect and link to each other in a manner similar to using grammatical information and grammatically related words to determine constraints from a larger context. Determining the link between two mentions in a text can take advantage of encyclopedic knowledge, and the system's ability to infer the connection between the mentions.

There is much work on unsupervised word sense disambiguation. Working with untagged data gives a system access to a much larger information base. Since selectional preferences acquired from sense-untagged corpora have worked well for the metonymy resolution task, we plan to push further towards unsupervised metonymy resolution, putting to use the lessons learned from unsupervised WSD.

Acknowledgements

We thank the Klaus Tschira Foundation, Heidelberg, for financial support, and the organizers of the SemEval-2007 Task #8 *Metonymy Resolution* – Katja Markert and Malvina Nissim – for making the annotated data freely available.

References

Caroline Brun, Maud Ehrmann, and Guillaume Jacquet. 2007. XRCE-M: A hybrid system for named entity metonymy resolution. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-1)*, Prague, Czech Republic, 23–24 June 2007, pages 488–491.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, Genoa, Italy, 22–28 May 2006, pages 449–454.

Richard Farkas, Eszter Simon, Gyorgy Szarvas, and Daniel Varga. 2007. GYDER: Maxent metonymy resolution. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-1)*, Prague, Czech Republic, 23–24 June 2007, pages 161–164.

Dan C. Fass. 1991. met*: A method for discriminating metonomy and metaphor by computer. *Computational Linguistics*, 17(1):49–90.

Sanda M. Harabagiu. 1998. Deriving metonymic coercions from WordNet. In *In Workshop on the Usage of WordNet in Natural Language Processing Systems*, Montreal, Canada, August 16, 1998, pages 142–148.

Jerry Hobbs, Mark Stickel, Douglas Appelt, and Paul Martin. 1993. Interpretation as abduction. *Artificial Intelligence*, 63(1-2):69–142.

Shin-ichiro Kamei and Takahiro Wakao. 1992. Metonymy: Reassessment, survey of acceptability, and its treatment in a machine translation system. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, Newark, Del., 28 June – 2 July 1992, pages 309–311.

Adam Kilgarriff, Pavel Rychly, Pavel Smrz, and David Tugwell. 2004. The Sketch Engine. In *Proceedings of the 11th International Congress of the European Association for Lexicography*, Lorient, France, 6–10 July 2004, pages 105–116.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, Madison, Wisc., 24–27 July 1998, pages 296–304.

Katja Markert and Udo Hahn. 2002. Metonymies in discourse. *Artificial Intelligence*, 135(1/2):145–198.

Katja Markert and Malvina Nissim. 2002. Metonymy resolution as classification task. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, Philadelphia, Penn., 6–7 July 2002, pages 204–213.

Katja Markert and Malvina Nissim. 2003. Corpus-based metonymy analysis. *Metaphor and Symbol*, 18(3):175–188.

Katja Markert and Malvina Nissim. 2007. SemEval-2007 Task 08: Metonymy Resolution at SemEval-2007. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-1)*, Prague, Czech Republic, 23–24 June 2007, pages 36–41.

Vivi Nastase and Michael Strube. 2008. Decoding Wikipedia category names for knowledge acquisition. In *Proceedings of the 23rd Conference on the Advancement of Artificial Intelligence*, Chicago, Ill., 13–17 July 2008, pages 1219–1224.

Vivi Nastase. 2008. Unsupervised all-words word sense disambiguation with grammatical dependencies. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing*, Hyderabad, India, 7–12 January 2008, pages 757–762.

Cristina Nicolae, Gabriel Nicolae, and Sanda Harabagiu. 2007. UTD-HLT-CG: Semantic architecture for metonymy resolution and classification of nominal relations. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-1)*, Prague, Czech Republic, 23–24 June 2007, pages 454–459.

Malvina Nissim and Katja Markert. 2003. Syntactic features and word similarity for supervised metonymy resolution. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, 7–12 July 2003, pages 56–63.

Malvina Nissim and Katja Markert. 2005. Learning to buy a Renault and talk to BMW: A supervised approach to conventional metonymy. In *Proceedings of the 6th International Workshop on Computational Semantics*, Tilburg, Netherlands, January 12-14, 2005.

James Pustejovsky. 1991. The generative lexicon. *Computational Linguistics*, 17(4):209–241.

Philip Resnik. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, (61):127–159.

David Stallard. 1993. Two kinds of metonymy. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio, 22–26 June 1993, pages 87–94.

Gustaf Stern. 1931. *Meaning and Changes of Meaning*. Indiana University Press, Bloomington, Indiana. (1968; first published in Sweden 1931).

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, Cal., 2nd edition.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivalling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, Mass., 26–30 June 1995, pages 189–196.

Segmenting Email Message Text into Zones

Andrew Lampert †‡
†CSIRO ICT Centre
PO Box 76
Epping 1710, Australia
andrew.lampert@csiro.au

Robert Dale ‡
rdale@ics.mq.edu.au

Cécile Paris †
‡Centre for Language Technology
Macquarie University
North Ryde 2109, Australia
cecile.paris@csiro.au

Abstract

In the early days of email, widely-used conventions for indicating quoted reply content and email signatures made it easy to segment email messages into their functional parts. Today, the explosion of different email formats and styles, coupled with the ad hoc ways in which people vary the structure and layout of their messages, means that simple techniques for identifying quoted replies that used to yield 95% accuracy now find less than 10% of such content. In this paper, we describe Zebra, an SVM-based system for segmenting the body text of email messages into nine zone types based on graphic, orthographic and lexical cues. Zebra performs this task with an accuracy of 87.01%; when the number of zones is abstracted to two or three zone classes, this increases to 93.60% and 91.53% respectively.

1 Introduction

Email message bodies consist of different functional parts such as email signatures, quoted reply content and advertising content. We refer to these as **email zones**. Many language processing tools stand to benefit from better knowledge of this message structure, facilitating focus on relevant content in specific parts of a message. In particular, access to zone information would allow email classification, summarisation and analysis tools to separate or filter out 'noise' and focus on the content in specific zones of a message that are relevant to the application at hand. Email contact mining tools such as that developed by Culotta et al. (2004), for example, might access the email signature zone, while tools that attempt to identify tasks or action items in email (e.g., (Bellotti et al., 2003; Corston-Oliver et al., 2004; Bennett and Carbonell, 2007; Lampert et al., 2007)) might restrict themselves to the sender-authored and forwarded content. Despite previous work on this problem, there are no available tools that can reliably extract or identify the different functional zones of an email message.

While there is no agreed standard set of email zones, there are clearly different functional parts within the body text of email messages. For example, the content of an email disclaimer is functionally different from the sender-authored content and from the quoted reply content automatically included from previous messages in the thread of conversation. Of course, there are different distinctions that can be drawn between zones; in this paper we explore several different categorisations based on our proposed set of nine underlying email zones.

Although we focus on content in the body of email messages, we recognise the presence of useful information in the semi-structured headers, and indeed make use of header information such as sender and recipient names in segmenting the unstructured body text.

Segmenting email messages into zones is a challenging task. Accurate segmentation is hampered by the lack of standard syntax used by different email clients to indicate different message parts, and by the ad hoc ways in which people vary the structure and layout of their messages. When replying to a message, for example, it is often useful to include all or part of the original message that is being replied to. Different email clients indicate quoted material in different ways. By default, some prefix every line of the quoted message with a character such as '>' or '|', while others indent the quoted content or insert the quoted message unmodified, prefixed by a message header. Sometimes the new content is above the quoted content (a style known as 'top-posting'); in other cases, the new content may appear after the quoted

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 919–928,
Singapore, 6-7 August 2009.

content (bottom-posting) or interleaved with the quoted content (inline replying). Confounding the issue further is that users are able to configure their email client to suit their individual tastes, and can change both the syntax of quoting and their quoting style (top, bottom or inline replying) on a per-message basis.

To address these challenges, in this paper we describe Zebra, our email zone classification system. First we describe how Zebra builds and improves on previous work in Section 2. Section 3 then presents our set of email zones, along with details of the email data we use for system training and experiments. In Section 4 we describe two approaches to zone classification, one that is line-based and one that is fragment-based. The performance of Zebra across two, three and nine email zone classification tasks is presented and analysed in Section 5.

2 Related Work

Segmenting email messages into zones requires both text segmentation and text classification. The main focus of most work on text segmentation is topic-based segmentation of news text (e.g., (Hearst, 1997; Beeferman et al., 1997)), but there have been some previous attempts at identifying functional zones in email messages.

Chen et al. (1999) looked at both linguistic and two-dimensional layout cues for extracting structured content from email signature zones in email messages. The focus of their work was on extracting information from already identified signature blocks using a combination of two-dimensional structural analysis and one-dimensional grammatical constraints; the intended application domain was as a component in a system for email text-to-speech rendering. The authors claim that their system can be modified to also identify signature blocks within email messages, but their system performs this task with a recall of only 53%. No attempt is made to identify functional zones other than email signatures.

Carvalho and Cohen's (2004) Jangada system attempted to identify email signatures within plain text email messages and to extract email signatures and reply lines. Unfortunately, the 20 Newsgroups corpus[1] they worked with contains 15-year-old Usenet messages which are much more homogeneous in their syntax than contemporary email, particularly in terms of how quoted text from previous messages is indicated. As a result, using a very simple metric (a line-initial '>' character) to identify reply lines achieves more than 95% accuracy. In contrast, this same simple metric applied to the Enron email data we annotated detects less than 10% of actual reply or forward lines.

Usenet messages are also markedly different from contemporary email when it comes to email signatures. Most Usenet clients produced messages which conformed to RFC3676 (Gellens, 2004), a standard that formalised a "long-standing convention in Usenet news ... of using two hyphens -- as the separator line between the body and the signature of a message." Unfortunately, this convention has long since ceased to be observed in email messages. Carvalho and Cohen's email signature detection approach also benefits greatly from a simplifying assumption that signatures are found in the last 10 lines of an email message. While this holds true for their Usenet message data, it is no longer the case for contemporary email.

In attempting to use Carvalho and Cohen's system to identify signature blocks and reply lines in our own work, we identified similar shortcomings to those noted by Estival et al. (2007). In particular, Jangada did not accurately identify forwarded or reply content in email data from the Enron email corpus. We believe that the use of older Usenet-style messages to train Jangada is a significant factor in the systematic errors the system makes in failing to identify quoted reply, forwarded and signature content in messages formatted in the range of message formats and styles popularised by Microsoft Outlook. These errors are a fundamental problem with Jangada, especially since Outlook is the most common client used to compose messages in our annotated email collection drawn from the Enron corpus. More generally, we note that Outlook is the most popular email client in current use, with an estimated 350–400 million users worldwide,[2] representing anywhere up to 40% of all email users.[3]

More recently, as part of their work on profiling

[1] http://people.csail.mit.edu/jrennie/20Newsgroups/

[2] Xobni Co-founder Adam Smith and former Engineering VP Gabor Cselle have both published Outlook user statistics. See http://www.xobni.com/asmith/archives/66 and http://gaborcselle.com/blog/2008/05/xobnis-journey-to-right-product.html.

[3] http://www.campaignmonitor.com/stats/email-clients/

authors of email messages, Estival et al. (2007) classified email bodies into five email zones. Their paper does not provide results for five-zone classification, but they report accuracy of 88.16% using a CRF classifier to distinguish three zones: reply, author and signature. We use their classification scheme as the starting point for our own set of email zones.

3 Email Zones

As noted earlier, we refer to the different functional components of email messages as **email zones**. The zones we propose refine and extend the five categories — *Author Text*, *Signature*, *Advertisement* (automatically appended advertising), *Quoted Text* (extended quotations such as song lyrics or poems), and *Reply Lines* (including forwarded and reply text) — identified by Estival et al. (2007).

We consider that each line of text in the body of an email message belongs to one of nine more fine-grained email zones. We intend our nine email zones to be abstracted and adapted to suit different tasks. To illustrate, we present the zones below abstracted into three classes: sender-authored content, boilerplate content, and content quoted from other conversations. This is the zone partition we use to generate the three-zone results reported in Section 5. This categorisation is useful for problems such as finding action items in email messages: such detection tools would look in text from the sender-authored message zones for new action item information, and could also look in quoted conversation content to link new action item information (such as reported completions) to previous action item content.

Our nine email zones can also be reduced to a binary scheme to distinguish text authored by the sender from text authored by others. This distinction is useful for problems such as author attribution or profiling tasks. In this two-class case, the sender-authored zones would be *Author*, *Greeting*, *Signoff* and *Signature*, while the other-authored zones would be *Reply*, *Forward*, *Disclaimer*, *Advertising* and *Attachment*. This is the partition of zones we use in our two-zone experiments reported in Section 5.

3.1 Sender Zones

Sender zones contain text written by the current email sender. The *Greeting* and *Signoff* zones are sub-zones of the *Author* zone, usually appearing as the first and last items respectively in the *Author* zone. Thus, our proposed sender zones are:

1. **Author:** New content from the current email sender. This specifically excludes any text authored by the sender that is included from previous messages.

2. **Greeting:** Terms of address and recipient names at the beginning of a message (e.g., *Dear/Hi/Hey Noam*).

3. **Signoff:** The message closing (e.g., *Thanks/Cheers/Regards, John*).

3.2 Quoted Conversation Zones

Quoted conversation zones include both content quoted in reply to previous messages in the same conversation thread and forwarded content from other conversations.[4] Our quoted conversation zones are:

4. **Reply:** Content quoted from a previous message in the same conversation thread, including any embedded signatures, attachments, advertising, disclaimers, author content and forwarded content. Content in a reply content zone may include previously sent content authored by the current sender.

5. **Forward:** Content from an email message outside the current conversation thread that has been forwarded by the current email sender, including any embedded signatures, attachments, advertising, disclaimers, author content and reply content.

3.3 Boilerplate Zones

Boilerplate zones contain content that is reused without modification across multiple email messages. Our proposed boilerplate zones are:

6. **Signature:** Content containing contact or other information that is automatically inserted in a message. In contrast to disclaimer or advertising content, signature content is usually templated content written once by the email author, and automatically or semi-automatically included in email messages. A

[4] Although we recognise the need for the *Quoted Text* zone proposed by Estival et al. (2007), no such data occurs in our collection of annotated email messages. We therefore omit this zone from our current set.

user may also use a *Signature* in place of a *Signoff*; in such cases, we still mark the text as a *Signature*.

7. **Advertising:** Advertising material in an email message. Such material often appears at the end of a message (e.g., *Do you Yahoo!?*), but may also appear prefixed or inline with the content of the message, (e.g., in sponsored mailing lists).

8. **Disclaimer:** Legal disclaimers and privacy statements, often automatically appended.

9. **Attachment:** Automated text indicating or referring to attached documents, such as that shown in line 16 of Figure 1. Note that this zone does not apply to manually authored reference to attachments, nor to the actual content of attachments (which we do not classify).

3.4 Email Data and Annotation

The training data for our zone classifier consists of 11881 annotated lines from almost 400 email messages drawn at random from the Enron email corpus (Klimt and Yang, 2004).[5] We use the database dump of the corpus released by Andrew Fiore and Jeff Heer.[6] This version of the corpus has been processed to remove duplicate messages and to normalise sender and recipient names, resulting in just over 250,000 email messages. No attachments are included. Following Estival et al. (2007), we used only a single annotator since the task revealed itself to be relatively uncontroversial. Each line in the body text of selected messages was marked by the annotator (one of the authors) as belonging to one of the nine zones. After removing blank lines, which we do not attempt to classify, we are left with 7922 annotated lines as training data for Zebra. The frequency of each zone within this annotated dataset is shown in Table 3.

Figure 1 shows an example of an email message with each line annotated with the appropriate email zone. Two zone annotations are shown for each line (in separate columns), one using the nine fine-grained zones and the second using the abstracted three-zone scheme described in Section 3. Note, however, that not all of the nine fine-grained zones, nor all of the three abstracted zones, are actually present in this particular message.

[5]This annotated dataset is available from http://zebra.thoughtlets.org/.

[6]http://bailando.sims.berkeley.edu/enron/enron.sql.gz

4 Zone Segmentation and Classification

Our email zone classification system is based around an SVM classifier using features that capture graphic, orthographic and lexical information about the content of an email message.

To classify the zones in an email message, we experimented with two approaches. The first employs a two-stage approach that segments a message into zone fragments and then classifies those fragments. Our second method simply classifies lines independently, returning a classification for each non-blank line in an email message. Our hypothesis was that classifying larger text fragments would lead to better performance due to the text fragments containing more cues about the zone type.

4.1 Zone Fragment Classification

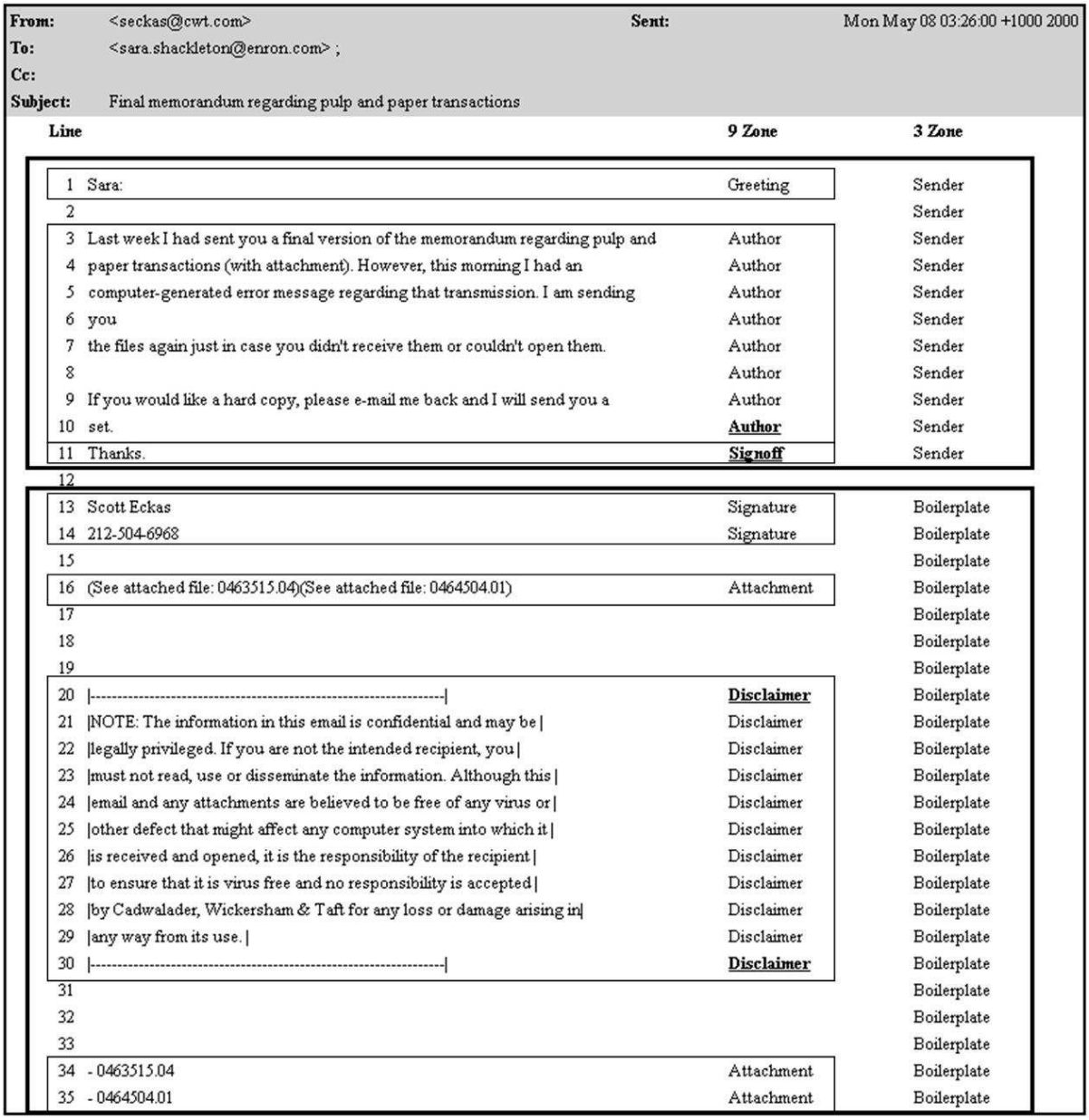
Zone fragment classification is a two-step process. First it predicts the zone boundaries using a simple heuristic, then it classifies the resulting *zone fragments*, the sets of content lines that lie between these hypothesised boundaries.

In order to determine how well we can detect zone boundaries, we first need to establish the correct zone boundaries in our collection of zone-annotated email messages.

4.1.1 Zone Boundaries

A zone boundary is defined as a continuous collection of one or more lines that separate two different email zones. Lines that separate two zones and are blank, contain only whitespace or contain only punctuation characters are called **buffer lines**.

Since classification of blank lines between zones is often ambiguous, empty or whitespace-only buffer lines are not included as content in any zone, and thus are not classified. Instead, they are treated as strictly part of the zone boundary. In Figure 1, these lines are shown without any zone annotation. Zone boundary lines that are included as content in a zone have their zone annotation styled in bold and underlined. The important point here is that zone boundaries are specific to a zone classification scheme. For nine-zone classification of the message in Figure 1, there are six zone boundaries: line 2, lines 10–11, line 12, line 15, lines 17–20, and lines 30–33. For three-zone clas-



From: <seckas@cwt.com> **Sent:** Mon May 08 03:26:00 +1000 2000
To: <sara.shackleton@enron.com> ;
Cc:
Subject: Final memorandum regarding pulp and paper transactions

Line		9 Zone	3 Zone
1	Sara:	Greeting	Sender
2			Sender
3	Last week I had sent you a final version of the memorandum regarding pulp and	Author	Sender
4	paper transactions (with attachment). However, this morning I had an	Author	Sender
5	computer-generated error message regarding that transmission. I am sending	Author	Sender
6	you	Author	Sender
7	the files again just in case you didn't receive them or couldn't open them.	Author	Sender
8		Author	Sender
9	If you would like a hard copy, please e-mail me back and I will send you a	Author	Sender
10	set.	**Author**	Sender
11	Thanks.	**Signoff**	Sender
12			
13	Scott Eckas	Signature	Boilerplate
14	212-504-6968	Signature	Boilerplate
15			Boilerplate
16	(See attached file: 0463515.04)(See attached file: 0464504.01)	Attachment	Boilerplate
17			Boilerplate
18			Boilerplate
19			Boilerplate
20	|----------|	**Disclaimer**	Boilerplate
21	|NOTE: The information in this email is confidential and may be|	Disclaimer	Boilerplate
22	|legally privileged. If you are not the intended recipient, you|	Disclaimer	Boilerplate
23	|must not read, use or disseminate the information. Although this|	Disclaimer	Boilerplate
24	|email and any attachments are believed to be free of any virus or|	Disclaimer	Boilerplate
25	|other defect that might affect any computer system into which it|	Disclaimer	Boilerplate
26	|is received and opened, it is the responsibility of the recipient|	Disclaimer	Boilerplate
27	|to ensure that it is virus free and no responsibility is accepted|	Disclaimer	Boilerplate
28	|by Cadwalader, Wickersham & Taft for any loss or damage arising in|	Disclaimer	Boilerplate
29	|any way from its use.|	Disclaimer	Boilerplate
30	|----------|	**Disclaimer**	Boilerplate
31			Boilerplate
32			Boilerplate
33			Boilerplate
34	- 0463515.04	Attachment	Boilerplate
35	- 0464504.01	Attachment	Boilerplate

Figure 1: An example email message marked with both nine- and three-zone annotations.

sification, the only zone boundary consists of line 12, separating the sender and boilerplate zones.

Based on these definitions, there are three different types of zone boundaries:

1. **Blank boundaries** contain only empty or whitespace-only buffer lines. Lines in these zone boundaries are strictly separate from the zone content. An example is Line 12 in Figure 1, for both the three- and nine-zone classification.
2. **Separator boundaries** contain only buffer lines, but must contain at least one punctuation-character buffer line that is retained as content in one or both zones. In Figure 1, an example is the zone boundary containing lines 17–20 that separates the *Attachment* and *Disclaimer* zones for nine-zone classification, since line 20 is retained as part of the *Disclaimer* zone content.
3. **Adjoining boundaries** consist of the last content line of the earlier zone and the first content line of the following zone. These boundaries occur where no buffer lines exist between the two zones. An example is the zone boundary containing lines 10 and 11 that separates the *Author* and *Signoff* zones in Figure 1 for nine-zone classification.

4.1.2 Hypothesising Zone Boundaries

To identify zone boundaries in unannotated email data, we employ a very simple heuristic approach. Specifically, we consider every line in the body of an email message that matches any of the following criteria to be a zone boundary:

1. A blank line;
2. A line containing only whitespace; or
3. A line beginning with four or more repeated punctuation characters, optionally prefixed by whitespace.

Our efforts to apply more sophisticated machine-learning techniques to identifying zone boundaries could not match the 90.15% recall achieved by this simple heuristic. The boundaries missed by the simple heuristic are all **adjoining boundaries**, where two zones are not separated by any buffer lines. An example of a boundary that is not detected by our heuristic is the zone boundary between the *Author* and *Signoff* zones in Figure 1 formed by lines 10 and 11.

Obviously, our simple boundary heuristic detects **actual boundaries** as well as **spurious boundaries** that do not actually separate different email zones. Unsurprisingly, the number of spurious boundaries is large. The precision of our simple heuristic across our annotated set of email messages is 22.5%, meaning that less than 1 in 4 hypothesised zone boundaries is an actual boundary. The underlying email zones average more than 12 lines in length, including just over 8 lines of non-blank content. Due to the number of spurious boundaries, fragments contain less than half this amount — approximately 3 lines of non-blank content on average. One of the most common types of spurious boundaries detected are the blank lines that frequently separate paragraphs within a single zone.

For three-zone classification, the set of predicted boundaries remains the same, but there are less actual boundaries to find, so recall increases to 96.3%. However, because many boundaries from the nine-zone classification are not boundaries for the three-zone classification, precision decreases to 14.7%.

4.1.3 Classifying Zone Fragments

Having segmented the email message into candidate zone fragments, we classify these fragments using the SMO implementation provided by Weka (Witten and Frank, 2005) with the features described in Section 4.3.

Although our boundary detection heuristic has better than 90% recall, the small number of actual boundaries that are not detected result in some zone fragments containing lines from more than one underlying email zone. In these cases, we consider the mode of all annotation values for lines in the fragment (i.e., the most frequent zone annotation) to be the gold-standard zone type for the fragment. This, of course, may mean that we somewhat unfairly penalise the accuracy of our automated classification when Zebra detects a zone that is indeed present in the fragment, but is not the most frequent zone.

4.2 Line Classification

Our line-based classification approach simply extracts all non-blank lines from an email message and classifies lines one-by-one, using the same features as for fragment-based classification. This approach is the same as the signature and reply line classification approach used by Carvalho and Cohen (2004).

4.3 Classification Features

We use a variety of graphic, orthographic and lexical features for classification in Zebra. The same features are applied in both the line-based and the fragment-based zone classification (to either individual lines or zone fragments). In the description of our features, we refer to both single lines and zone fragments (collections of contiguous lines) as **text fragments**.

4.3.1 Graphic Features

Our graphic features capture information about the presentation and layout of text in an email message, independent of the actual words used. This information is a crucial source of information for identifying zones. Such information includes how the text is organised and ordered, as well as the 'shape' of the text. The specific features we employ are:

- the number of words in the text fragment;
- the number of Unicode code points (i.e., characters) in the text fragment;
- the start position of the text fragment (equal to one for the first line in the message, two for the second line and increasing monotonically

through the message; we also normalise the result for message length);
- the end position of the text fragment (calculated as above and again normalised for message length);
- the average line length (in characters) within the text fragment (equal to the line length for line-based text fragments);
- the length of the text fragment (in characters) relative to the previous fragment;
- the length of the text fragment (in characters) relative to the following fragment;
- the number of blank lines preceding the text fragment; and
- the number of blank lines following the text fragment.

4.3.2 Orthographic Features

Our orthographic features capture information about the use of distinctive characters or character sequences including punctuation, capital letters and numbers. Like our graphic features, orthographic features tend to be independent of the words used in an email message. The specific orthographic features we employ include:

- whether all lines start with the same character (e.g., '>');
- whether a prior text fragment in the message contains a quoted header;
- whether a prior text fragment in the message contains repeated punctuation characters;
- whether the text fragment contains a URL;
- whether the text fragment contains an email address;
- whether the text fragment contains a sequence of four or more digits;
- the number of capitalised words in the text fragment;
- the percentage of capitalised words in the text fragment;
- the number of non-alpha-numeric characters in the text fragment;
- the percentage of non-alpha-numeric characters in the text fragment;
- the number of numeric characters in the text fragment;
- the percentage of numeric characters in the text fragment;
- whether the message subject line contains a reply syntax marker such as *Re:* ; and
- whether the message subject line contains a forward syntax marker such as *Fw:*.

4.3.3 Lexical Features

Finally, our lexical features capture information about the words used in the email text. We use unigrams to capture information about the vocabulary and word bigram features to capture short range word order information. More specifically, the lexical features we apply to each text fragment include:

- each word unigram, calculated with a minimum frequency threshold cutoff of three, represented as a separate binary feature;
- each word bigram, calculated with a minimum frequency threshold cutoff of three, represented as a separate binary feature;
- whether the text fragment contains the sender's name;
- whether a prior text fragment in the message contains the sender's name;
- whether the text fragment contains the sender's initials; and
- whether the text fragment contains a recipient's name.

Features that look for instances of sender or recipient names are less likely to be specific to a particular business or email domain. These features use regular expressions to find name occurrences, based on semi-structured information in the email message headers. First, we extract and normalise the names from the email headers to identify the relevant person's given name and surname. Our features then capture whether one or both of the given name or surname are present in the current text fragment. Features which detect user initials make use of the same name normalisation code to retrieve a canonical form of the user's name, from which their initials are derived.

5 Results and Discussion

Table 1 shows Zebra's accuracy in classifying email zones. The results are calculated using 10-fold cross-validation. Accuracy is shown for three tasks — nine-, three- and two-zone classification — using both line and zone-fragment classification. Performance is compared against a majority class baseline in each case.

Zebra's performance compares favourably with previously published results. While it is difficult to

	2 Zones		3 Zones		9 Zones	
	Zebra	Baseline	Zebra	Baseline	Zebra	Baseline
Lines	93.60%	61.14%	91.53%	58.55%	87.01%	30.94%
Fragments	92.09%	62.18%	91.37%	59.44%	86.45%	30.36%

Table 1: Classification accuracy compared against a majority baseline

	2 Zones		3 Zones		9 Zones	
	Zebra	Baseline	Zebra	Baseline	Zebra	Baseline
Lines	90.62%	61.14%	86.56%	58.55%	81.05%	30.94%
Fragments	91.14%	62.18%	89.44%	59.44%	82.55%	30.36%

Table 2: Classification accuracy, without word n-gram features, compared against a majority baseline

directly compare, since not all systems are freely available and they are not trained or tested over the same data, our three-zone classification (identifying sender, boilerplate and quoted reply content) is very similar to the three-zone task for which (Estival et al., 2007) report 88.16% accuracy for their system and 64.22% accuracy using Carvalho and Cohen's Jangada system. Zebra outperforms both, achieving 91.53% accuracy using a line-based approach. In the two-zone task, where we attempt to identify sender-authored lines, Zebra achieves 93.60% accuracy and an F-measure of 0.918, exceeding the 0.907 F-measure reported for Estival et al.'s system tuned for exactly this task.

Interestingly, the line-based approach provides slightly better performance than the fragment-based approach for each of the two-zone, three-zone and nine-zone classification tasks. As noted earlier, our original hypothesis was that zone fragments would contain more information about the sequence and text shape of the original message, and that this would lead to better performance for fragment-based classification.

When we restrict our feature set to those that look only at the text of the line or zone fragment, the fragment-based approach does perform better than the line-based one. Using only word unigram features, for example, our fragment classifier achieves 78.7% accuracy. Using the same features, the line-based classifier achieves only 57.5% accuracy. When we add further features that capture sequence and shape information from outside the text fragment being classified (e.g., the length of a text segment compared to the text segment before and after, and whether a segment occurs after another segment containing repeated punctuation or the sender's name), the line-based approach achieves a greater increase in accuracy than the fragment-based approach. This presumably is because individual lines intrinsically have less information about the message context, and so benefit more from the information added by the new features.

We also experimented with removing all word unigram and bigram features to explore the classifier's portability across different domains. This removed all vocabulary and word order information from our feature set. In doing so, our feature set was reduced to less than thirty features, consisting of mostly graphic and orthographic information. The few remaining lexical features captured only the presence of sender and recipient names, which are independent of any particular email domain. As expected, performance did drop, but not dramatically. Table 2 shows that average performance without n-grams (across two-, three- and nine-zone tasks) for line-based classification drops by 4.67%. In contrast, fragment-based classification accuracy drops by less than half this amount — an average of 2.26%. This suggests that, as we originally hypothesised, there are additional non-lexical cues in zone fragments that give information about the zone type. This makes the zone fragment approach potentially more portable for use across email data from different enterprise domains.

Of course, classification accuracy gives only a limited picture of Zebra's performance. Table 4 shows precision and recall results for each zone in the nine-zone line-based classification task. Per-

	Total	Author	Signature	Disclaim	Advert	Greet	Signoff	Reply	Fwd	Attach
Author	2415	2197	56	9	4	14	31	43	53	8
Signature	383	93	203	4	0	0	20	28	31	4
Disclaim	97	30	4	52	0	0	0	2	9	0
Advert	83	47	1	1	20	0	0	7	7	0
Greet	85	8	0	0	0	74	2	0	1	0
Signoff	195	30	5	0	0	0	147	11	2	0
Reply	2451	49	10	3	2	1	10	2222	154	0
Fwd	2187	72	13	7	8	1	3	125	1958	0
Attach	26	4	0	0	0	0	0	1	1	20

Table 3: Confusion Matrix for 9 Zone Line Classification

formance clearly varies significantly across the different zones. For *Author*, *Greeting*, *Reply* and *Forward* zones, performance is good, with F-measure > 0.8. This is encouraging, given that many email tools, such as action-item detection and email summarisation would benefit from an ability to separate author content from reply content and forwarded content. The *Advertising*, *Signature* and *Disclaimer* zones show the poorest performance, particularly in terms of Recall. The *Advertising* and *Disclaimer* zones are almost certainly hindered by a lack of training data; they are two of the smallest zones in terms of number of lines of training data. The relatively poor *Signature* class performance is more interesting. Given the potential confusion between *Signoff* content and *Signature*s that function as *Signoff*s, one might expect confusion between *Signoff* and *Signature* zones, but Table 3 shows this is not the case. Instead, there is significant confusion between *Signature* and *Author* content, with almost 25% of *Signature* lines misclassified as *Author* lines. When word n-grams are removed from the feature set, the number of these misclassifications increases to almost 50%. These results reinforce our observation that the task of email signature extraction is much more difficult that it was in the days of Usenet messages.

Zone	Precision	Recall	F-Measure
Author	0.868	0.910	0.889
Signature	0.695	0.530	0.601
Disclaimer	0.684	0.536	0.601
Advertising	0.588	0.241	0.342
Greeting	0.822	0.871	0.846
Signoff	0.690	0.754	0.721
Reply	0.911	0.907	0.909
Forward	0.884	0.895	0.889
Attachment	0.625	0.769	0.690

Table 4: Precision and recall for nine-zone line classification

6 Conclusion

Identifying functional zones in email messages is a challenging task, due in large part to the diversity in syntax used by different email software, and the dynamic manner in which people employ different styles in authoring email messages. Zebra, our system for segmenting and classifying email message text into functional zones, achieves performance that exceeds comparable systems, and that is at a level to be practically useful to email researchers and system builders. In addition to releasing our annotated email dataset, the Zebra system will also be available for others to use[7].

Because we employ a non-sequential learning algorithm, we encode sequence information into the feature set. In future work, we plan to determine the effectiveness of using a sequential learning algorithm like Conditional Random Fields (CRF). We note, however, that Carvalho and Cohen (2004) demonstrate that using a non-sequential learning algorithm with sequential features, as we do, has the potential to meet or exceed the performance of sequential learning algorithms.

Acknowledgments

The authors are grateful to the anonymous reviewers for their insightful comments and suggestions.

[7] See http://zebra.thoughtlets.org for access to the annotated data and Zebra system

References

Douglas Beeferman, Adam Berger, and John Lafferty. 1997. Text segmentation using exponential models. In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing*, pages 35–46, Providence, RI.

Victoria Bellotti, Nicolas Ducheneaut, Mark Howard, and Ian Smith. 2003. Taking email to task: The design and evaluation of a task management centred email tool. In *Computer Human Interaction Conference*, CHI, pages 345–352, Ft Lauderdale, Florida, USA, April 5-10.

Paul N Bennett and Jaime G Carbonell. 2007. Combining probability-based rankers for action-item detection. In *Proceedings of NAACL HLT 2007*, pages 324–331, Rochester, NY, April.

Vitor R Carvalho and William W Cohen. 2004. Learning to extract signature reply lines from email. In *Proceedings of First Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, July 30-31.

Hao Chen, Jianying Hu, and Richard W Sproat. 1999. Integrating geometrical and linguistic analysis for email signature block parsing. *ACM Transactions on Information Systems*, 17(4):343–366, October. ISSN: 1046-8188.

Simon H. Corston-Oliver, Eric Ringger, Michael Gamon, and Richard Campbell. 2004. Task-focused summarization of email. In *ACL-04 Workshop: Text Summarization Branches Out*, pages 43–50, July.

Aron Culotta, Ron Bekkerman, and Andrew McCallum. 2004. Extracting social networks and contact information from email and the web. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*.

Dominique Estival, Tanja Gaustad, Son Bao Pham, Will Radford, and Ben Hutchinson. 2007. Author profiling for English emails. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 263–272, Melbourne, Australia, Sept 19-21.

R. Gellens. 2004. RFC3676: The text/plain format and delsp parameters, February.

Marti A. Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.

Bryan Klimt and Yiming Yang. 2004. Introducing the Enron corpus. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*.

Andrew Lampert, Cécile Paris, and Robert Dale. 2007. Can requests-for-action and commitments-to-act be reliably identified in email messages? In *Proceedings of the 12th Australasian Document Computing Symposium*, pages 48–55, Melbourne, Australia, December 10.

Ian Witten and Eiba Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition.

Hypernym Discovery Based on Distributional Similarity and Hierarchical Structures

Ichiro Yamada[†], Kentaro Torisawa[†], Jun'ichi Kazama[†], Kow Kuroda[†], Masaki Murata[†], Stijn De Saeger[†], Francis Bond[†] and Asuka Sumida[‡]

[†]National Institute of Information and Communications Technology
3-5 Hikaridai, Keihannna Science City 619-0289, JAPAN
{iyamada,torisawa,kazama,kuroda,murata,stijn,bond}@nict.go.jp

[‡]Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi-shi, Ishikawa-ken 923-1211, JAPAN
a-sumida@jaist.ac.jp

Abstract

This paper presents a new method of developing a large-scale hyponymy relation database by combining Wikipedia and other Web documents. We attach new words to the hyponymy database extracted from Wikipedia by using distributional similarity calculated from documents on the Web. For a given target word, our algorithm first finds *k* similar words from the Wikipedia database. Then, the hypernyms of these *k* similar words are assigned scores by considering the distributional similarities and hierarchical distances in the Wikipedia database. Finally, new hyponymy relations are output according to the scores. In this paper, we tested two distributional similarities. One is based on raw verb-noun dependencies (which we call "RVD"), and the other is based on a large-scale clustering of verb-noun dependencies (called "CVD"). Our method achieved an attachment accuracy of 91.0% for the top 10,000 relations, and an attachment accuracy of 74.5% for the top 100,000 relations when using CVD. This was a far better outcome compared to the other baseline approaches. Excluding the region that had very high scores, CVD was found to be more effective than RVD. We also confirmed that most relations extracted by our method cannot be extracted merely by applying the well-known lexico-syntactic patterns to Web documents.

1 Introduction

Large-scale taxonomies such as WordNet (Fellbaum 1998) play an important role in information extraction and question answering. However, extremely high costs are borne to manually enlarge and maintain such taxonomies. Thus, applications using these taxonomies tend to face the drawback of data sparseness. This paper presents a new method for discovering a large set of hyponymy relations. Here, a word[1] X is regarded as a hypernym of a word Y if Y is a kind of X or Y is an instance of X. We are able to generate large-scale hyponymy relations by attaching new words to the hyponymy database extracted from Wikipedia (referred to as "Wikipedia relation database") by using distributional similarity calculated from Web documents. Relations extracted from Wikipedia are relatively clean. On the other hand, reliable distributional similarity can be calculated using a large number of documents on the Web. In this paper, we combine the advantages of these two resources.

Using distributional similarity, our algorithm first computes *k* similar words for a target word. Then, each *k* similar word assigns a score to its ancestors in the hierarchical structures of the Wikipedia relation database. The hypernym that has the highest score for the target word is selected as the hypernym of the target word. Figure 1 is an overview of the proposed approach.

In the experiment, we extracted hypernyms for approximately 670,000 target words that are not included in the Wikipedia relation database but are found on the Web. We tested two distributional similarities: one based on raw verb-noun dependencies (RVD) and the other based on a large-scale clustering of verb-noun dependencies (CVD). The experimental results showed that the proposed methods were more effective than the other baseline approaches. In addition, we confirmed that most of the relations extracted by our method could not be extracted using the lexico-syntactic pattern-based method.

In the remainder of this paper, we first intro-

[1] In this paper, we use the term "word" for both "a single-word word" and "a multi-word word."

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 929–937,
Singapore, 6-7 August 2009.

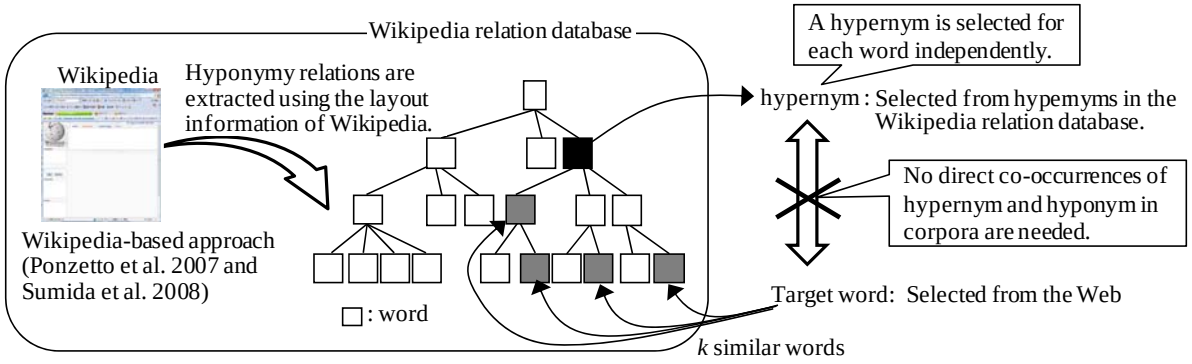


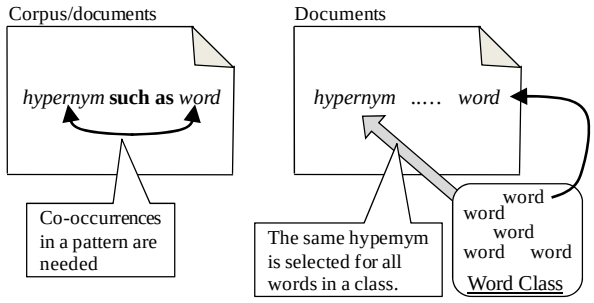
Figure 1: Overview of the proposed approach.

duce some related works in Section 2. Section 3 describes the Wikipedia relation database. Section 4 describes the distributional similarity calculated by the two methods. In Section 5, we describe a method to discover an appropriate hypernym for each target word. The experimental results are presented in Section 6 before concluding the paper in Section 7.

2 Related Works

Most previous researchers have relied on lexico-syntactic patterns for hyponymy acquisition. Lexico-syntactic patterns were first used by Hearst (1992). The patterns used by her included "NP_0 such as NP_1," in which NP_0 is a hypernym of NP_1. Using these patterns as seeds, Hearst discovered new patterns by which to semi-automatically extract hyponymy relations. Pantel et al. (2004a) proposed a method to automatically discover the patterns using a minimal edit distance. Ando et al. (2003) applied predefined lexico-syntactic patterns to Japanese news articles. Snow et al. (2005) generalized these lexico-syntactic pattern-based methods by using dependency path features for machine learning. Then, they extended the framework such that this method was capable of making use of heterogenous evidence (Snow et al. 2006). These pattern-based methods require the co-occurrences of a target word and the hypernym in a document. It should be noted that the requirement of such co-occurrences actually poses a problem when we extract a large set of hyponymy relations since they are not frequently observed (Shinzato et al. 2004, Pantel et al. 2004b).

Clustering-based methods have been proposed as another approach. Caraballo (1999), Pantel et al. (2004b), and Shinzato et al. (2004) proposed a method to find a common hypernym for word classes, which are automatically constructed using some measures of word similarities or hierarchical structures in HTML documents. Etzioni et al. (2005) used both a pattern-based approach and a clustering-based approach. The required amount of co-occurrences is significantly reduced due to class-based generalization processes. Note that these clustering-based methods obtain the same hypernym for all the words in a particular class. This causes a problem for selecting an appropriate hypernym for each word in the case when the granularity or the construction of the classes is incorrect. Figure 2 shows the drawbacks of the existing approaches.



Pattern-based method (Hearst 1992, Pantel et al. 2004a, Ando et al. 2003, Snow et al. 2005, Snow et al. 2006, and Etzioni et al. 2005)

Clustering-based method (Caraballo 1999, Pantel et al. 2004b, Shinzato et al. 2004, and Etzioni et al. 2005)

Figure 2: Drawbacks in existing approaches for hyponymy acquisition.

Ponzetto et al. (2007) and Sumida et al. (2008) proposed a method for acquiring hyponymy relations from Wikipedia. This Wikipedia-based approach can extract a large volume of hyponymy relations with high accuracy. However, it is also true that this approach does not account for many words that usually appear in Web documents; this could be because of the unbalanced topics in Wikipedia or merely because of the incomplete coverage of articles on Wikipedia. Our method can target words that frequently appear on the Web but are not included in the Wikipedia relation database, thus making the results of the Wikipedia-based approach richer and more balanced. Our approach uses distributional similari-

ty, which is computed based on the noun-verb dependency profiles on the Web. The use of distributional similarity resembles the clustering-based approach; however, our method can select a hypernym for each word independently, and it does not suffer from class granularity mismatch or the low quality of classes. In addition, our approach exploits the hierarchical structures of the Wikipedia hypernym relations.

3 Wikipedia Relation Database

Our Wikipedia relation database is based on the extraction method of Sumida et al. (2008). They proposed a method of automatically acquiring hyponymy relations by focusing on the hierarchical layout of articles on Wikipedia. By way of an example, Figure 3 shows part of the source code clipped from the article titled "Penguin." An article has hierarchical structures composed of titles, sections, itemizations, etc. The entire article is divided into sections titled "Anatomy," "Mating habits," "Systematics and evolution," "Penguins in popular culture," and so on. The section "Systematics and evolution" has a subsection "Systematics," which is further divided into "Aptenodytes," "Eudyptes," and so on. Some of these section-subsection relations can be regarded as valid hyponymy relations. In this article, relations such as the one between "Aptenodytes" and "Emperor Penguin" and that between "Book" and "Penguins of the World" are valid hyponymy relations.

First, Sumida et al. (2008) extracted hyponymy relation candidates from hierarchical structures on Wikipedia. Then, they selected proper hyponymy relations using a support vector machine classifier. They used several kinds of features for the hyponymy relation candidate, such as a POS tag for each word, the appearance of morphemes of each word, the distance between two words in the hierarchical structures of Wikipedia, and the last character of each word. As a result of their experiments, approximately 2.4 million hyponymy relations in Japanese were extracted, with a precision rate of 90.1%.

Compared to the traditional taxonomies, these extracted hyponymy relations have the following characteristics (Fellbaum 1998, Bond et al. 2008).

(a) The database includes a more extensive vocabulary.
(b) The database includes a large number of named entities.

Popular Japanese taxonomies *GoiTaikei* (Ikehara et al. 1997) and *Bunrui-Goi-Hyo* (1996)

```
'''Penguins''' are a group of
[[Aquatic animal|aquatic]],
[[flightless bird]]s.
== Anatomy ==
== Mating habits ==
==Systematics and evolution==
===Systematics===
* Aptenodytes
**[[Emperor Penguin]]
** [[King Penguin]]
* Eudyptes
== Penguins in popular culture ==
== Book ==
* Penguins
* Penguins of the World
== Notes ==
* Penguinone
* the [[Penguin missile]]
[[Category:Penguins]]
[[Category:Birds]]
```

Figure 3: A part of source code clipped from the article "Penguin" in Wikipedia.

contain approximately 300,000 words and 96,000 words, respectively. In contrast, the extracted hyponymy relations contain approximately 1.2 million hyponyms and are undoubtedly much larger than the existing taxonomies. Another difference is that since Wikipedia covers a large number of named entities, the extracted hyponymy relations also contain a large number of named entities.

Note that the extracted relations have a hierarchical structure because one hypernym of a certain word may also be the hyponym of another hypernym. However, we observed that the depth of the hierarchy, on an average, is extremely shallow. To make the hierarchy appropriate for our method, we extended these into a deeper hierarchical structure. The extracted relations include many compound nouns as hypernyms, and we decomposed a compound noun into a sequence of nouns using a morphological analyzer. Since Japanese is a head-final language, the suffix of a noun sequence becomes the hypernym of the original compound noun if the suffix forms another valid compound noun. We extracted suffixes of compound nouns and manually checked whether they were valid compound nouns; then, we constructed a hierarchy of compound nouns. The hierarchy can be extended such that it includes the hyponyms of the original hypernym and the resulting hierarchy constitutes a hierarchical taxonomy. We use this hierarchical taxonomy as a target for expansion.[2]

[2] Note that this modification was performed as part of another project of ours aimed at constructing a large-scale and clean hypernym knowledge base by human annotation. We do not think this cost is directly relevant to the method proposed here.

4 Distributional Similarity

The distributional hypothesis states that words that occur in similar contexts tend to be semantically similar (Harris 1985). In this section, we first introduce distributional similarity based on raw verb-noun dependencies (RVD). To avoid the sparseness problem of the co-occurrence of verb-noun dependencies, we also use distributional similarity based on a large-scale clustering of verb-noun dependencies (CVD).

In the experiment mentioned in the following section, we used the TSUBAKI corpus (Shinzato et al. 2008) to calculate distributional similarity. This corpus provides a collection of 100 million Japanese Web pages containing 6×10^9 sentences.

4.1 Distributional Similarity Based on RVD

When calculating the distributional similarity based on RVD, we use the triple <*v*, *rel*, *n*>, where *v* is a verb, *n* is a noun phrase, and *rel* stands for the relation between *v* and *n*. In Japanese, a relation *rel* is represented by postpositions attached to *n* and the phrase composed of *n* and *rel* modifies *v*. Each triple is divided into two parts. The first is <*v*, *rel*> and the second is *n*. Then, we consider the conditional probability of occurrence of the pair <*v*, *rel*>: *P*(<*v*, *rel*>|*n*). *P*(<*v*, *rel*>|*n*) can be regarded as the distribution of the grammatical contexts of the noun phrase *n*. The distributional similarity can be defined as the distance between these distributions. There are several kinds of functions for evaluating the distance between two distributions (Lee 1999). Our method uses the Jensen-Shannon divergence. The Jensen-Shannon divergence between two probability distributions, $P(\cdot \mid n_1)$ and $P(\cdot \mid n_2)$, can be calculated as follows:

$$\begin{aligned} & D_{JS}(P(\cdot \mid n_1) \,\|\, P(\cdot \mid n_2)) \\ & = \frac{1}{2}(D_{KL}(P(\cdot \mid n_1) \,\|\, \frac{P(\cdot \mid n_1) + P(\cdot \mid n_2)}{2}) \\ & + D_{KL}(P(\cdot \mid n_2) \,\|\, \frac{P(\cdot \mid n_1) + P(\cdot \mid n_2)}{2})), \end{aligned}$$

where D_{KL} indicates the Kullback-Leibler divergence and is defined as follows:

$$D_{KL}(P(\cdot \mid n_1) \,\|\, P(\cdot \mid n_2)) = \sum P(\cdot \mid n_1) \log \frac{P(\cdot \mid n_1)}{P(\cdot \mid n_2)}.$$

Finally, the distributional similarity between two words, n_1 and n_2, is defined as follows:

$$sim(n_1, n_2) = 1 - D_{JS}(P(\cdot \mid n_1) \,\|\, P(\cdot \mid n_2)).$$

This similarity assumes a value from 0 to 1. If two words are similar, the value will be close to 1; if two words have entirely different meanings, the value will be 0.

In the experiment, we used 1,000,000 noun phrases and 100,000 pairs of verbs and postpositions to calculate the probability *P*(<*v*, *rel*>|*n*) from the dependency relations extracted from the above-mentioned Web corpus (Shinzato et al. 2008). The probabilities are computed using the following equation by modifying for the frequency using the log function:

$$P(< v, rel > \mid n) = \frac{\log(f(< v, rel, n >)) + 1}{\sum_{< v, rel > \in D} \log(f(< v, rel, n >) + 1} \quad \text{if } f(< v, rel, n >) > 0,$$

where *f*(<*v*, *rel*, *n*>) is the frequency of a triple <*v*, *rel*, *n*> and *D* is the set defined as { <*v*, *rel* > | *f*(<*v*, *rel*, *n*>) > 0 }. In the case of *f*(<*v*, *rel*, *n*>) = 0, P(<*v*, *rel*>|*n*) is set to 0.

Instead of using the observed frequency directly as in the usual maximum likelihood estimation, we modified it as above. Although this might seems strange, this kind of modification is common in information retrieval as a term weighing method (Manning et al. 1999) and it is also applied in some studies to yield better word similarities (Terada et al. 2006, Kazama et al. 2009). We also adopted this idea in this study.

4.2 Distributional Similarity Based on CVD

Rooth et al. (1999) and Torisawa (2001) showed that EM-based clustering using verb-noun dependencies can produce semantically clean noun clusters. We exploit these EM-based clustering results as the smoothed contexts for noun *n*. In Torisawa's model (2001), the probability of occurrence of the triple <*v*, *rel*, *n*> is defined as follows:

$$\begin{aligned} & P(< v, rel, n >) \\ & \quad =_{def} \sum_{a \in A} P(< v, rel > \mid a) P(n \mid a) P(a), \end{aligned}$$

where *a* denotes a hidden class of <*v*,*rel*> and *n*. In this equation, the probabilities *P*(<*v*,*rel*>|*a*), *P*(*n*|*a*), and *P*(*a*) cannot be calculated directly because class *a* is not observed in a given corpus. The EM-based clustering method estimates these probabilities using a given corpus. In the E-step,

the probability $P(a|<v,rel>)$ is calculated. In the M-step, the probabilities $P(<v,rel>|a)$, $P(n|a)$, and $P(a)$ are updated to arrive at the maximum likelihood using the results of the E-step. From the results of estimation of this EM-based clustering method, we can obtain the probabilities $P(<v,rel>|a)$, $P(n|a)$, and $P(a)$ for each $<v, rel>$, n, and a. Then, $P(a|n)$ is calculated by the following equation:

$$P(a \mid n) = \frac{P(n \mid a)P(a)}{\sum_{a \in A} P(n \mid a)P(a)}.$$

$P(a|n)$ can be used to find the class of n. For example, the class that has the maximum $P(a|n)$ can be regarded as the class to which n belongs. Noun phrases that occur with similar pairs $<v,rel>$ tend to be classified in the same class.

Kazama et al. (2008) proposed the parallelization of this EM-based clustering with the aim of enabling large-scale clustering and using the resulting clusters in named entity recognition. Kazama et al. (2009) reported the calculation of distributional similarity using the clustering results. The distributional similarity was calculated by the Jensen-Shannon divergence, which was used in this paper. Similar to the case in Kazama et al., we performed word clustering using 1,000,000 noun phrases and 2,000 classes. Note that the frequencies of dependencies were modified with the log function, as in RVD, described in the previous section.

5 Discovering an Appropriate Hypernym for a Target word

In the Wikipedia relation database, there are about 95,000 hypernyms and about 1.2 million hyponyms. In both RVD and CVD, the words used were selected according to the number (the number of kinds, not the frequency) of $<v, rel>$s that n has dependencies in the data. As a result, 1 million words were selected. The number of common words that are also included in the Wikipedia relation database are as follows:

Hypernyms	28,015 (common hypernyms)
Hyponyms	175,022 (common hyponyms)

These common hypernyms become candidates for hypernyms for a target word. On the other hand, the common hyponyms are used as clues for identifying appropriate hypernyms.

In our task, the potential target words are about 810,000 in number and are not included in the Wikipedia relation database. These include some strange words or word phrases that are extracted due to the failure of morphological analysis. We exclude these words using simple rules. Consequently, the number of target words for our process is reduced to about 670,000.

In the following section, we outline the scoring method that uses k similar words to discover an appropriate hypernym for a target word. We also explain several baseline approaches that use distributional similarity.

5.1 Scoring with *k* similar Words

In this approach, we first calculate the similarities between the common hyponyms and a target word and select the k most similar common hyponyms. Here, we use a similarity threshold value S_{min} to avoid the effect of words having lower similarities. If the similarity is less than the threshold value, the word is excluded from the set of k similar words. Next, each k similar word votes a score to its ancestors in the hierarchical structures of the Wikipedia relation database. The score used to vote for a hypernym n_{hyper} is as follows:

$$score(n_{hyper}) = \sum_{n_{hypo} \in Desc(n_{hyper}) \cap ksimilar(n_{trg})} d^{r(n_{hyper}, n_{hypo})-1} \times sim(n_{trg}, n_{hypo}),$$

where n_{trg} is the target word, $Desc(n_{hyper})$ is the descendant of the hypernym n_{hyper}, $ksimilar(n_{trg})$ is the k similar word of n_{trg}, $d^{r(n_{hyper}, n_{hypo})-1}$ is a penalty that depends on the differences in the depth of hierarchy, d is a parameter for the penalty value and has a value between 0 and 1, and $r(n_{trg}, n_{hypo})$ is the difference in the depth of hierarchy between n_{trg} and n_{hypo}. $sim(n_{trg}, n_{hypo})$ is a distributional similarity between n_{trg} and n_{hypo}.

As a result of scoring, each hypernym has a score for the target word. The hypernym that has the highest score for the target word is selected as its hypernym. The hyponymy relations thus produced are ranked according to the scores.

Figure 4 shows an example of the scoring process. In this example, we use *CitroenAX* as the target word whose hypernym will be identified. First, the k similar words are extracted from the common hyponyms in the Wikipedia relation: *Opel Astra*, *TVR Tuscan*, *Mitsubishi Minica*, and *Renault Lutecia* are extracted. Next, each k similar word votes a score to its ancestors. The words *Opel Astra*, *TVR Tuscan*, and *Renault Lutecia* vote to their parent *car* and the word *Mitsubishi*

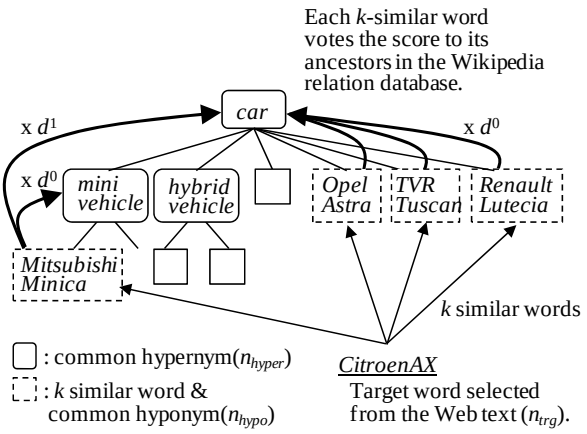


Figure 4: Overview of the scoring process.

Minica votes to its parent *mini-vehicle* and its grandparent *car* with a small penalty. Finally, the hypernym *car*, which has the highest score, is selected as the hypernym of the target word *CitroenAX*.

5.2 Baseline Approaches

Using distributional similarity, we can also develop the following baseline approaches to discover hyponymy relations.

Selecting the hypernym of the most similar hyponym (baseline approach 1)

We use the heuristics that similar words tend to have the same hypernym. In this approach, we first calculate the similarities between the common hyponyms and the target word. The common hyponym most similar to the target word is extracted. Then, the parent of the extracted common hyponym is regarded as the hypernym of the target word. This approach outputs several hypernyms when the most similar hyponym has several hypernyms. This approach can be considered to be the same as the scoring method using k similar words when $k = 1$. We use the distributional similarity between the target word and the most similar hyponym in the Wikipedia relation database as the score for the appropriateness of the resulting hyponymy.

Selecting the most similar hypernym (baseline approach 2)

The distributional similarity between the common hypernym and the target word is calculated. Then, the hypernym that has the highest distributional similarity is regarded as the hypernym of the target word. The similarity is used as the score of the appropriateness of the produced hyponymy.

Scoring based on the average similarity of the hypernym's children (baseline approach 3)

This approach uses the probabilistic distributions of the hypernym's children. We define the probability $P_{child}(\cdot \mid n_{hyper})$ characterized by the children of the hypernym n_{hyper}, as follows:

$$P_{child}(\cdot \mid n_{hyper}) = \frac{\sum_{n_{hypo} \in Ch(n_{hyper})} P(\cdot \mid n_{hypo}) P(n_{hypo})}{\sum_{n_{hypo} \in Ch(n_{hyper})} P(n_{hypo})},$$

where $Ch(n_{hyper})$ is a set of all children of n_{hyper}. Then, distributional similarities between a common hypernym n_{hyper} and the target word n_{hypo} are calculated. The hypernym that has the highest distributional similarity is selected as the hypernym of the word. This distributional similarity is used as the score of the appropriateness of the produced hyponymy.

If a hypernym has only a few children, the reliability of the probabilistic distribution of hypernym defined here will be low because the Wikipedia relation database includes some incorrect relations. For this reason, we use the hypernym only if the number of children it has is more than a threshold value.

6 Experiments

We evaluated our proposed methods by using it in experiments to discover hypernyms from the Wikipedia relation database for the target words extracted from about 670,000 noun phrases.

6.1 Parameter Estimation by Preliminary Experiments

In the proposed methods, there are several parameters. We performed parameter optimization by randomly selecting 694 words as development data in our preliminary experiments. The hypernyms of these words were determined manually. We adjusted the parameters so that each method achieved the best performance for this development data.

The parameters in the scoring method with k similar words were adjusted as follows[3]:

(RVD)

Number of similar words:	$k = 100$.
Similarity threshold:	$S_{min} = 0.05$.
Penalty value for ancestors:	$d = 0.6$.

[3] We tested the parameter values k = {100, 200, 300, 400, 500, 600, 700, 800, 900, 1000}, S_{min}={0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4} and d={0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.0}.

Table 1: Precision of each approach based on the score ranking. CVD represents the method that uses the distributional similarity based on large-scale of clustering of verb-noun dependencies. RVD represents the one based on raw verb-noun dependencies.

	k-similar words (CVD)	*k*-similar words (RVD)	*k*-similar words (CVD, $d = 0$)	Baseline approach 1 (CVD)	Baseline approach 2 (CVD)	Baseline approach 3 (CVD)
1,000	0.940	**1.000**	0.850	0.730	0.290	0.630
10,000	**0.910**	0.875	0.875	0.555	0.300	0.445
100,000	**0.745**	0.710	0.730	0.500	0.280	0.435
670,000	**0.520**	0.500	0.470	0.345	0.115	0.170

(CVD)

Number of similar words: $k = 200$.
Similarity threshold: $S_{min} = 0.3$.
Penalty value for ancestors: $d = 0.6$.

The parameter in baseline approach 3 was adjusted as follows:

Threshold for the number of children: 20.

6.2 Evaluation of the Experimental Results on the Basis of Score Ranking

Using the adjusted parameters, we conducted experiments to extract the hypernym of each target word with the help of the scoring method based on *k* similar words. In these experiments, two kinds of distributional similarity mentioned in Section 4 were exploited individually. The words that were used in the development data were excluded.

We also conducted a comparative experiment in which the parameter value for the penalty of the hierarchal difference, *d*, was set to 0 to clarify the ability of using hierarchal structures in the *k* similar words method. This means each *k* similar word votes only to their parent.

We then judged the quality of each acquired hypernym. The evaluation data sets were sampled from the top 1,000, 10,000, 100,000, and 670,000 results that were ranked according to the score of each method. Then, against 200 samples that were randomly sampled from each set, one of the authors judged whether the hypernym extracted by each method for the target word was correct or not. In this evaluation, if the sentence "*The target word* is a kind of *the hypernym*" or "*The target word* is an instance of *the hypernym*" was consistent, the extracted hyponymy was judged as correct. It should be noted that the outputs of the compared methods are combined and shuffled to enable fair comparison. In addition, baseline approach 1 extracted several hypernyms for the target word. In this case, we judged the hypernym as correct when the case where one of the hypernyms was correct.

The precision of each result is shown in Table 1. The results of the *k* similar words method are far better than those of the other baseline methods. In particular, the *k* similar words method with CVD outperformed the methods of the *k* similar words where the parameter value *d* was set to 0 and the method using RVD except for the top 1,000 results. This means that the use of hierarchal structures and the clustering process for calculating distributional similarity are effective for this task. We confirmed the significant differences of the proposed method (CVD) as compared with all the baseline approaches at the 1% significant level by the Fisher's exact test (Hays 1988).

The precision of baseline approach 2 that selected the most similar hypernym was the worst among all the methods. There were words that were similar to the target word among the hypernyms extracted incorrectly. For example, the word *semento-kojo* (cement factory) was extracted for the hypernym of the word *kuriningu-kojo* (dry cleaning plant). It is difficult to judge whether the word is a hypernym or just a similar word by using only the similarity measure.

As for the results of baseline approach 1 using the most similar hyponym and baseline approach 3 using the similarity of the set of hypernym's children, the noise on the Wikipedia relation database decreased the precision. Moreover, over-specified hypernyms were extracted incorrectly by these methods. In contrast, the method of scoring based on the use of *k* similar words was robust against noise because it uses the voting approach for the similarities. Further, this method can extract hypernyms that are not over-specific because it uses all descendants for scoring.

Table 2 shows some examples of relations extracted by the *k* similar words method using CVD.

Table2: Hypernym discovery results by the *k*-similar words based approach (CVD). The underline indicates the hypernyms which are extracted incorrectly.

Score	Target word	Extracted hypernym
58.6	INDIVI	*burando* (fashion label)
54.3	*kureome* (Cleome)	*hana* (flower)
34.4	UOKR	*gemu* (game)
21.7	*Okido* (Okido)	*machi* (town)
20.5	*Sumatofotsu* (Smart fortwo)	*kuruma* (car)
15.6	*Fukagawameshi* (Fukagawa rice)	*ryori* (dish)
8.9	John Barry	*sakkyokuka* (composer)
8.5	JVM	*sofuto-wea* (software)
6.6	*metangasu* (methane gas)	*genso* (chemical element)
5.4	*me-ru semina* (mail seminar)	*Hon* (book)
3.9	*gurometto* (grommet)	*shohin* (merchandise)
3.1	*supuringubakku* (spring back)	*gensho* (phenomenon)

6.3 Investigation of the Extracted Relation Overlap with a Conventional Method

We randomly sampled 300 hyponymy relations that were extracted correctly using the *k* similar words method exploiting CVD and investigated whether or not these relations can be extracted by the conventional method based on the lexico-syntactic pattern. The possible hyponymy relations were extracted using the pattern-based method (Ando et al. 2003) from the TSUBAKI corpus (Shinzato et al. 2008). From a comparison of these relations, we found only 57 common hyponymy relations. That is, the remaining 243 hyponymy relations were not included in the possible hyponymy relations. This result indicates that our method can acquire the hyponymy relations that cannot be extracted by the conventional pattern-based method.

6.4 Discussions

We investigated the reason for the errors generated by the method of scoring using *k* similar words exploiting CVD. We conducted experiments on hypernym extraction targeting 694 words in the development data mentioned in Section 6.1. Among these, 286 relations were extracted incorrectly. In these relations, there were some frequent hypernyms. For example, the word *sakuhin* (work) appeared 28 times and *hon* (book) appeared 20 times. As shown in Table 2, *hon* (book) was also extracted for the target word *meru-seminah* (mail seminar). It is really difficult even for a human to identify whether the title is that of the book or the event. If we can identify these difficult hypernyms in advance, we can improve precision by excluding them from the target hypernyms. This will be one of the topics for future study.

7 Conclusion

In this paper, we proposed a method for discovering hyponymy relations between nouns by fusing the Wikipedia relation database and words from the Web. We demonstrated that the method using *k* similar words has high accuracy. The experimental results showed the effectiveness of using hierarchal structures and the clustering process for calculating distributional similarity for this task. The experimental results showed that our method could achieve 91.0% attachment accuracy for the top 10,000 hyponymy relations and 74.5% attachment accuracy for the top 100,000 relations when using the clustering-based similarity. We confirmed that most relations extracted by the proposed method could not be handled by the lexico-syntactic pattern-based method. Future work will be to filter out difficult hypernyms for hyponymy extraction process to achieve higher precision.

References

M. Ando, S. Sekine and S. Ishizaki. 2003. Automatic Extraction of Hyponyms from Newspaper Using Lexicosyntactic Patterns. *IPSJ SIG Notes*, 2003-NL-157, pp. 77–82 (in Japanese).

F. Bond, H. Isahara, K. Kanzaki and K. Uchimoto. 2008. Boot-strapping a WordNet Using Multiple Existing WordNets. In *the 6th International Conference on Language Resources and Evaluation* (LREC), Marrakech.

Bunruigoihyo. 1996. The National Language Research Institute (in Japanese).

S. A. Caraballo. 1999. Automatic Construction of a Hypernym-labeled Noun Hierarchy from Text. In *Proceedings of the Conference of the Association for Computational Linguistics* (ACL).

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld and A. Yates. 2005. Unsupervised Named-Entity Extraction from the Web: An Experimental Study. *Artificial Intelligence*, 165(1):91–134.

C. Fellbaum. 1998. WordNet: An Electronic Lexical

Database. Cambridge, MA: MIT Press.

Z. Harris. 1985. Distributional Structure. In Katz, J. J. (ed.) The Philosophy of Linguistics, *Oxford University Press*, pp. 26–47.

W. L. Hays. 1988. Statistics: Analyzing Qualitative Data, Rinehart and Winston, Inc., Ch. 18, pp. 769–783.

M. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th Conference on Computational Linguistics* (COLING), pp. 539–545.

S. Ikehara, M. Miyazaki, S. Shirai, A. Yokoo, H. Nakaiwa, K. Ogura, Y. Ooyama and Y. Hayashi. 1997. Goi-Taikei A Japanese Lexicon, *Iwanami Shoten.*

J. Kazama and K. Torisawa. 2008. Inducing Gazetteers for Named Entity Recognition by Large-scale Clustering of Dependency Relations. In *Proceedings of ACL-08: HLT*, pp. 407–415.

J. Kazama, Stijn De Saeger, K. Torisawa and M. Murata. 2009. Generating a Large-scale Analogy List Using a Probabilistic Clustering Based on Noun-Verb Dependency Profiles. In *15th Annual Meeting of the Association for Natural Language Processing*, C1–3 (in Japanese).

L. Lee. 1999. Measures of Distributional Similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 25–32.

C. D. Manning and H. Schutze. 1999. Foundations of Statistical Natural Language Processing. The MIT Press.

P. Pantel, D. Ravichandran and E. Hovy. 2004a. Towards Terascale Knowledge Acquisition. In *Proceedings of the 20th International Conference on Computational Linguistics*.

P. Pantel and D. Ravichandran. 2004b. Automatically Labeling Semantic Classes. In *Proceedings of the Human Language Technology and North American Chapter of the Association for Computational Linguistics Conference*.

S. P. Ponzetto, and M. Strube. 2007. Deriving a Large Scale Taxonomy from Wikipedia. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, pp. 1440–1445.

M. Rooth, S. Riezler, D. Presher, G. Carroll and F. Beil. 1999. Inducing a Semantically Annotated Lexicon via EM-based Clustering. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pp. 104–111.

K. Shinzato and K. Torisawa. 2004. Acquiring Hyponymy Relations from Web Documents. In *Proceedings of HLT-NAACL*, pp. 73–80.

K. Shinzato, D. Kawahara, C. Hashimoto and S. Kurohashi. 2008. A Large-Scale Web Data Collection as A Natural Language Processing Infrastructure. In *the 6th International Conference on Language Resources and Evaluation* (LREC).

R. Snow, D. Jurafsky and A. Y. Ng. 2005. Learning Syntactic Patterns for Automatic Hypernym Discovery. *NIPS 2005*.

R. Snow, D. Jurafsky, A. Y. Ng. 2006. Semantic Taxonomy Induction from Heterogenous Evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pp. 801–808.

A. Sumida, N. Yoshinaga and K. Torisawa. 2008. Boosting Precision and Recall of Hyponymy Relation Acquisition from Hierarchical Layouts in Wikipedia. In *the 6th International Conference on Language Resources and Evaluation* (LREC).

A. Terada, M. Yoshida, H. Nakagawa. 2006. A Tool for Constructing a Synonym Dictionary using context Information. In *proceedings of IPSJ SIG Technical Reports*, vol.2006 No.124, pp. 87-94. (In Japanese).

K. Torisawa. 2001. An Unsupervised Method for Canonicalization of Japanese Postpositions. In *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium* (NLPRS), pp. 211–218.

K. Torisawa, Stijn De Saeger, Y. Kakizawa, J. Kazama, M. Murata, D. Noguchi and A. Sumida. 2008. TORISHIKI-KAI, An Autogenerated Web Search Directory. In *Proceedings of the second international symposium on universal communication*, pp. 179–186, 2008.

Web-Scale Distributional Similarity and Entity Set Expansion

Patrick Pantel[†], Eric Crestan[†], Arkady Borkovsky[‡], Ana-Maria Popescu[†], Vishnu Vyas[†]

[†]Yahoo! Labs
Sunnyvale, CA 94089
{ppantel,ecrestan}@yahoo-inc.com
{amp,vishnu}@yahoo-inc.com

[‡]Yandex Labs
Burlingame, CA 94010
arkady@yandex-team.ru

Abstract

Computing the pairwise semantic similarity between all words on the Web is a computationally challenging task. Parallelization and optimizations are necessary. We propose a highly scalable implementation based on distributional similarity, implemented in the MapReduce framework and deployed over a 200 billion word crawl of the Web. The pairwise similarity between 500 million terms is computed in 50 hours using 200 quad-core nodes. We apply the learned similarity matrix to the task of automatic set expansion and present a large empirical study to quantify the effect on expansion performance of corpus size, corpus quality, seed composition and seed size. We make public an experimental testbed for set expansion analysis that includes a large collection of diverse entity sets extracted from Wikipedia.

1 Introduction

Computing the semantic similarity between terms has many applications in NLP including word classification (Turney and Littman 2003), word sense disambiguation (Yuret and Yatbaz 2009), context-spelling correction (Jones and Martin 1997), fact extraction (Paşca et al. 2006), semantic role labeling (Erk 2007), and applications in IR such as query expansion (Cao et al. 2008) and textual advertising (Chang et al. 2009).

For commercial engines such as Yahoo! and Google, creating lists of named entities found on the Web is critical for query analysis, document categorization, and ad matching. Computing term similarity is typically done by comparing co-occurrence vectors between all pairs of terms (Sarmento et al. 2007). Scaling this task to the Web requires parallelization and optimizations.

In this paper, we propose a large-scale term similarity algorithm, based on distributional similarity, implemented in the MapReduce framework and deployed over a 200 billion word crawl of the Web. The resulting similarity matrix between 500 million terms is applied to the task of expanding lists of named entities (automatic set expansion). We provide a detailed empirical analysis of the discovered named entities and quantify the effect on expansion accuracy of corpus size, corpus quality, seed composition, and seed set size.

2 Related Work

Below we review relevant work in optimizing similarity computations and automatic set expansion.

2.1 Computing Term Similarities

The distributional hypothesis (Harris 1954), which links the meaning of words to their contexts, has inspired many algorithms for computing term similarities (Lund and Burgess 1996; Lin 1998; Lee 1999; Erk and Padó 2008; Agirre et al. 2009). Brute force similarity computation compares all the contexts for each pair of terms, with complexity $O(n^2m)$ where n is the number of terms and m is the number of possible contexts. More efficient strategies are of three kinds:

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 938–947,
Singapore, 6-7 August 2009.

Smoothing: Techniques such as *Latent Semantic Analysis* reduce the context space by applying truncated Singular Value Decomposition (SVD) (Deerwester et al. 1990). Computing the matrix decomposition however does not scale well to web-size term-context matrices. Other currently unscalable smoothing techniques include Probabilistic Latent Semantic Analysis (Hofmann 1999), Iterative Scaling (Ando 2000), and Latent Dirichlet Allocation (Blei et al. 2003).

Randomized Algorithms: Randomized techniques for approximating various similarity measures have been successfully applied to term similarity (Ravichandran et al. 2005; Gorman and Curran 2006). Common techniques include *Random Indexing* based on Sparse Distributed Memory (Kanerva 1993) and *Locality Sensitive Hashing* (Broder 1997).

Optimizations and Distributed Processing: Bayardo et al. (2007) present a sparse matrix optimization strategy capable of efficiently computing the similarity between terms which's similarity exceeds a given threshold. Rychlý and Kilgarriff (2007), Elsayed et al. (2008) and Agirre et al. (2009) use reverse indexing and the MapReduce framework to distribute the similarity computations across several machines. Our proposed approach combines these two strategies and efficiently computes the exact similarity (cosine, Jaccard, Dice, and Overlap) between *all* pairs.

2.2 Entity extraction and classification

Building entity lexicons is a task of great interest for which structured, semi-structured and unstructured data have all been explored (GoogleSets; Sarmento et al. 2007; Wang and Cohen 2007; Bunescu and Mooney 2004; Etzioni et al. 2005; Paşca et al. 2006). Our own work focuses on set expansion from *unstructured* Web text. Apart from the choice of a data source, state-of-the-art entity extraction methods differ in their use of *numerous*, *few* or *no* labeled examples, the *open* or *targeted* nature of the extraction as well as the types of features employed. Supervised approaches (McCallum and Li 2003, Bunescu and Mooney 2004) rely on large sets of labeled examples, perform targeted extraction and employ a variety of sentence- and corpus-level features. While very precise, these methods are typically used for coarse grained entity classes (*People, Organizations, Companies*) for which large training data sets are available. Unsupervised approaches rely on no labeled data and use either bootstrapped class-specific extraction patterns (Etzioni et al. 2005) to find new elements of a given class (for targeted extraction) or corpus-based term similarity (Pantel and Lin 2002) to find term clusters (in an open extraction framework). Finally, semi-supervised methods have shown great promise for identifying and labeling entities (Riloff and Shepherd 1997; Riloff and Jones 1999; Banko et al. 2007; Downey et al. 2007; Paşca et al. 2006; Paşca 2007a; Paşca 2007b; Paşca and Durme 2008). Starting with a set of seed entities, semi-supervised extraction methods use either class-specific patterns to populate an entity class or *distributional similarity* to find terms similar to the seed set (Paşca's work also examines the advantages of combining these approaches). Semi-supervised methods (including ours) are useful for extending finer grain entity classes, for which large unlabeled data sets are available.

2.3 Impact of corpus on system performance

Previous work has examined the effect of using large, sometimes Web-size corpora, on system performance in the case of familiar NLP tasks. Banko and Brill (2001) show that Web-scale data helps with confusion set disambiguation while Lapata and Keller (2005) find that the Web is a good source of n-gram counts for unsupervised models. Atterer and Schutze (2006) examine the influence of corpus size on combining a supervised approach with an unsupervised one for relative clause and PP-attachment. Etzioni et al. (2005) and Pantel et al. (2004) show the advantages of using large quantities of generic Web text over smaller corpora for extracting relations and named entities. Overall, corpus size and quality are both found to be important for extraction. Our paper adds to this body of work by focusing on the task of similarity-based set expansion and providing a large empirical study quantify the relative corpus effects.

2.4 Impact of seeds on extraction performance

Previous extraction systems report on the size and quality of the training data or, if semi-supervised, the size and quality of entity or pattern seed sets. Narrowing the focus to closely related work, Paşca (2007a; 2007b) and Paşca and Durme (2008) show the impact of varying the number of instances representative of a given class and the size of the attribute seed set on the precision of class attribute extraction. An example observation is that good

Table 1. Definitions for f_0, f_1, f_2, and f_3 for commonly used similarity scores.

METRIC	$f_0(x,y,z)$	$f_1(x,y)$	$f_2(\vec{x}) = f_3(\vec{x})$
Overlap	x	1	0
Jaccard*	$\frac{x}{y+z-x}$	$\min(x,y)$	$\sum_i x_i$
Dice*	$\frac{2x}{y+z}$	$x \times y$	$\sum_i x_i^2$
Cosine	$\frac{x}{y \times z}$	$x \times y$	$\sqrt{\sum_i x_i^2}$

*weighted generalization

quality class attributes can still be extracted using 20 or even 10 instances to represent an entity class. Among others, Etzioni et al. (2005) shows that a small pattern set can help bootstrap useful entity seed sets and reports on the impact of seed set noise on final performance. Unlike previous work, empirically quantifying the influence of seed set size and quality on extraction performance of random entity types is a key objective of this paper.

3 Large-Scale Similarity Model

Term semantic models normally invoke the distributional hypothesis (Harris 1985), which links the meaning of terms to their contexts. Models are built by recording the surrounding contexts for each term in a large collection of unstructured text and storing them in a term-context matrix. Methods differ in their definition of a context (e.g., *text window* or *syntactic relations*), or by a means to weigh contexts (e.g., *frequency*, *tf-idf*, *pointwise mutual information*), or ultimately in measuring the similarity between two context vectors (e.g., using *Euclidean distance*, *Cosine*, *Dice*).

In this paper, we adopt the following methodology for computing term similarity. Our various web crawls, described in Section 6.1, are POS-tagged using Brill's tagger (1995) and chunked using a variant of the Abney chunker (Abney 1991). Terms are NP chunks with some modifiers removed; their contexts (i.e., features) are defined as their rightmost and leftmost stemmed chunks. We weigh each context f using pointwise mutual information (Church and Hanks 1989). Let PMI*(w)* denote a pointwise mutual information vector, constructed for each term as follows: $PMI(w) = (pmi_{w1}, pmi_{w2}, \ldots, pmi_{wm})$, where pmi_{wf} is the pointwise mutual information between term w and feature f:

$$pmi_{wf} = \log \frac{c_{wf} \times N}{\sum_{i=1}^{n} c_{if} \times \sum_{j=1}^{m} c_{wj}}$$

where c_{wf} is the frequency of feature f occurring for term w, n is the number of unique terms and N is the total number of features for all terms.

Term similarities are computed by comparing these *pmi* context vectors using measures such as cosine, Jaccard, and Dice.

3.1 Large-Scale Implementation

Computing the similarity between terms on a large Web crawl is a non-trivial problem, with a worst case cubic running time – $O(n^2m)$ where n is the number of terms and m is the dimensionality of the feature space. Section 2.1 introduces several optimization techniques; below we propose an algorithm for large-scale term similarity computation which calculates *exact* scores for *all* pairs of terms, generalizes to several different metrics, and is scalable to a large crawl of the Web.

Our optimization strategy follows a generalized sparse-matrix multiplication approach (Sarawagi and Kirpal 2004), which is based on the well-known observation that a scalar product of two vectors depends only on the coordinates for which both vectors have non-zero values. Further, we observe that most commonly used similarity scores for feature vectors $\vec{x}$ and $\vec{y}$, such as *cosine* and *Dice*, can be decomposed into three values: one depending only on features of $\vec{x}$, another depending only on features of $\vec{y}$, and the third depending on the features shared both by $\vec{x}$ and $\vec{y}$. More formally, commonly used similarity scores $F(\vec{x},\vec{y})$ can be expressed as:

$$F(\vec{x},\vec{y}) = f_0\left(\sum_i f_1(x_i,y_i), f_2(\vec{x}), f_3(\vec{y})\right)$$

Table 1 defines f_0, f_1, f_2, and f_3 for some common similarity functions. For each of these scores, $f_2 = f_3$. In our work, we compute all of these scores, but report our results using only the *cosine* function.

Let A and B be two matrices of PMI feature vectors. Our task is to compute the similarity between all vectors in A and all vectors in B. In computing the similarity between all pairs of terms, $A = B$.

Figure 1 outlines our algorithm for computing the similarity between all elements of A and B. Efficient computation of the similarity matrix can be achieved by leveraging the fact that $F(\vec{x},\vec{y})$ is determined solely by the features shared by $\vec{x}$ and $\vec{y}$ (i.e., $f_1(0,x) = f_1(x,0) = 0$ for any x) and that most of

the feature vectors are very sparse (i.e., most possible contexts never occur for a given term). In this case, calculating $f_1(x, y)$ is only required when both feature vectors have a shared non-zero feature, significantly reducing the cost of computation. Determining which vectors share a non-zero feature can easily be achieved by first building an inverted index for the features. The computational cost of this algorithm is $\sum N_i^2$, where N_i is the number of vectors that have a non-zero i^{th} coordinate. Its worst case time complexity is $O(ncv)$ where n is the number of terms to be compared, c is the maximum number of non-zero coordinates of any vector, and v is the number of vectors that have a non-zero i^{th} coordinate where i is the coordinate which is non-zero for the most vectors. In other words, the algorithm is efficient only when the density of the coordinates is low. On our datasets, we observed near linear running time in the corpus size.

Bayardo et al. (2007) described a strategy that potentially reduces the cost even further by omitting the coordinates with the highest number of non-zero value. However, their algorithm gives a significant advantage only when we are interested in finding solely the similarity between highly similar terms. In our experiments, we compute the exact similarity between all pairs of terms.

Distributed Implementation

The pseudo-code in Figure 1 assumes that A can fit into memory, which for large A may be impossible. Also, as each element of B is processed independently, running parallel processes for non-intersecting subsets of B makes the processing faster. In this section, we outline our MapReduce implementation of Figure 1 deployed using Hadoop[1], the open-source software package implementing the MapReduce framework and distributed file system. Hadoop has been shown to scale to several thousands of machines, allowing users to write simple "map" and "reduce" code, and to seamlessly manage the sophisticated parallel execution of the code. A good primer on MapReduce programming is in (Dean and Ghemawat 2008).

Our implementation employs the MapReduce model by using the *Map* step to start $M \times N$ *Map* tasks in parallel, each caching $1/M$th part of A as an inverted index and streaming $1/N$th part of B through it. The actual inputs are read by the tasks directly from HDFS (Hadoop Distributed File System). Each part of A is processed N times, and each part of B is processed M times. M is determined by the amount of memory dedicated for the inverted index, and N should be determined by trading off the fact that as N increases, more parallelism can be obtained at the increased cost of building the same inverse index N times.

Input: Two matrices A and B of feature vectors.

```
## Build an inverted index for A (optimiza-
## tion for data sparseness)
AA = an empty hash-table
for i in (1..n):
   F2[i] = f2(A[i]) ## cache values of f2(x)
   for k in non-zero features of A[i]:
      if k not in AA: AA[k] = empty-set
      ## append <vector-id, feature-value>
      ## pairs to the set of non-zero
      ## values for feature k
      AA[k].append( (i,A[i,k]) )
## Process the elements of B
for b in B:
   F1 = {} ## the set of Ai that have non-
        zero similarity with b
   for k in non-zero features of b:
      for i in AA[k]:
         if i not in sim: sim[i] = 0
         F1[i] += f1( AA[k][i], b[k])
   F3 = f3(b)
   for i in sim:
      print i, b, f0( F1[i], F2[i], F3)
```

Output: A matrix containing the similarity between all elements in A and in B.

Figure 1. Similarity computation algorithm.

The similarity algorithm from Figure 1 is run in each task of the *Map* step of a MapReduce job. The *Reduce* step is used to group the output by b_i.

4 Application to Set Expansion

Creating lists of named entities is a critical problem at commercial engines such as Yahoo! and Google. The types of entities to be expanded are often not known a priori, leaving supervised classifiers undesirable. Additionally, list creators typically need the ability to expand sets of varying granularity. Semi-supervised approaches are predominantly adopted since they allow targeted expansions while requiring only small sets of seed entities. State-of-the-art techniques first compute term-term similarities for all available terms and then select candidates for set expansion from amongst the terms most similar to the seeds (Sarmento et al. 2007).

[1] Hadoop, http://lucene.apache.org/hadoop/

Formally, we define our expansion task as:

> **Task Definition**: Given a set of seed entities $S = \{s_1, s_2, ..., s_k\}$ of a class $C = \{s_1, s_2, ..., s_k, ..., s_n\}$ and an unlabeled textual corpus T, find all members of the class C.

For example, consider the class of *Bottled Water Brands*. Given the set of seeds S = {*Volvic*, *San Pellegrino*, *Gerolsteiner Brunnen, Bling* H^2O}, our task is to find all other members of this class, such as {*Agua Vida*, *Apenta, Culligan*, *Dasani*, *Ethos Water*, *Iceland Pure Spring Water*, *Imsdal*, …}

4.1 Set Expansion Algorithm

Our goal is not to propose a new set expansion algorithm, but instead to test the effect of using our Web-scale term similarity matrix (enabled by the algorithm proposed in Section 3) on a state-of-the-art distributional set expansion algorithm, namely (Sarmento et al. 2007).

We consider S as a set of prototypical examples of the underlying entity set. A representation for the meaning of S is computed by building a feature vector consisting of a weighted average of the features of its seed elements $s_1, s_2, ..., s_k$, a centroid. For example, given the seed elements {*Volvic*, *San Pellegrino*, *Gerolsteiner Brunnen, Bling* H^2O}, the resulting centroid consists of (details of the feature extraction protocol are in Section 6.1):

```
brand, mineral water, monitor,
lake, water, take over, …
```

Centroids are represented in the same space as terms allowing us to compute the similarity between centroids and all terms in our corpus. A scored and ranked set for expansion is ultimately generated by sorting all terms according to their similarity to the seed set centroid, and applying a cutoff on either the similarity score or on the total number of retrieved terms. In our reported experiments, we expanded over 22,000 seed sets using our Web similarity model from Section 3.

5 Evaluation Methodology

In this section, we describe our methodology for evaluating Web-scale set expansion.

5.1 Gold Standard Entity Sets

Estimating the quality of a set expansion algorithm requires a random sample from the universe of all entity sets that may ever be expanded, where a set represents some concept such as *Stage Actors*. An approximation of this universe can be extracted from the "*List of*" pages in Wikipedia[2].

Upon inspection of a random sample of the "*List of*" pages, we found that several lists were compositions or joins of concepts, for example "*List of World War II aces from Denmark*" and "*List of people who claimed to be God*". We addressed this issue by constructing a quasi-random sample as follows. We randomly sorted the list of every noun occurring in Wikipedia[2]. Then, for each noun we verified whether or not it existed in a Wikipedia list, and if so we extracted this list. If a noun belonged to multiple lists, the authors chose the list that seemed most appropriate. Although this does not generate a perfect random sample, diversity is ensured by the random selection of nouns and relevancy is ensured by the author adjudication.

The final gold standard consists of 50 sets, including: *classical pianists*, *Spanish provinces*, *Texas counties*, *male tennis players*, *first ladies*, *cocktails*, *bottled water brands*, and *Archbishops of Canterbury*. For each set, we then manually scraped every instance from Wikipedia keeping track also of the listed variants names.

The gold standard is available for download at:

http://www.patrickpantel.com/cgi-bin/Web/Tools/getfile.pl?type=data&id=sse-gold/wikipedia.20071218.goldsets.tgz

The 50 sets consist on average of 208 instances (with a minimum of 11 and a maximum of 1,116) for a total of 10,377 instances.

5.2 Trials

In order to analyze the corpus and seed effects on performance, we created 30 copies of each of the 50 sets and randomly sorted each copy. Then, for each of the 1500 copies, we created a trial for each of the following 23 seed sizes: 1, 2, 5, 10, 20, 30, 40, …, 200. Each trial of seed size s was created by taking the first s entries in each of the 1500 random copies. For sets that contained fewer than 200 items, we only generated trials for seed sizes

[2] In this paper, extractions from Wikipedia are taken from a snapshot of the resource in December 2008.

Table 2. Corpora used to build our expansion models.

CORPORA	UNIQUE SENTENCES (MILLIONS)	TOKENS (MILLIONS)	UNIQUE WORDS (MILLIONS)
Web100	5,201	217,940	542
Web020†	1040	43,588	108
Web004†	208	8,717	22
Wikipedia[6]	30	721	34

†Estimated from *Web100* statistics.

smaller than the set size. The resulting trial dataset consists of 20,220 trials[3].

5.3 Judgments

Set expansion systems consist of an expansion algorithm (such as the one described in Section 4.1) as well as a corpus (such as Wikipedia, a news corpus, or a web crawl). For a given system, each of the 20,220 trials described in the previous section are expanded. In our work, we limited the total number of system expansions, per trial, to 1000.

Before judgment of an expanded set, we first collapse each instance that is a variant of another (determined using the variants in our gold standard) into one single instance (keeping the highest system score)[4]. Then, each expanded instance is judged as *correct* or *incorrect* automatically against the gold standard described in Section 5.1.

5.4 Analysis Metrics

Our experiments in Section 6 consist of precision vs. recall or precision vs. rank curves, where:

a) ***precision*** is defined as the percentage of correct instances in the expansion of a seed set; and

b) ***recall*** is defined as the percentage of non-seed gold standard instances retrieved by the system.

Since the gold standard sets vary significantly in size, we also provide the *R*-precision metric to normalize for set size:

c) ***R-precision*** is defined as the average precision of all trials where precision is taken at rank R = {size of trial's associated gold standard set}, thereby normalizing for set size.

For the above metrics, 95% confidence bounds are computed using the randomly generated samples described in Section 5.2.

6 Experimental Results

Our goal is to study the performance gains on set expansion using our Web-scale term similarity algorithm from Section 3. We present a large empirical study quantifying the importance of corpus and seeds on expansion accuracy.

6.1 Experimental Setup

We extracted statistics to build our model from Section 3 using four different corpora, outlined in Table 2. The *Wikipedia* corpus consists of a snapshot of the *English* articles in December 2008[5]. The *Web100* corpus consists of an extraction from a large crawl of the Web, from Yahoo!, of over 600 million English webpages. For each crawled document, we removed paragraphs containing fewer than 50 tokens (as a rough approximation of the narrative part of a webpage) and then removed all duplicate sentences. The resulting corpus consists of over 200 billion words. The *Web020* corpus is a random sample of $1/5^{th}$ of the sentences in *Web100* whereas *Web004* is a random sample of $1/25^{th}$ of *Web100*.

For each corpus, we tagged and chunked each sentence as described in Section 3. We then computed the similarity between all noun phrase chunks using the model of Section 3.1.

6.2 Quantitative Analysis

Our proposed optimization for term similarity computation produces *exact* scores (unlike randomized techniques) for *all* pairs of terms on a large Web crawl. For our largest corpus, *Web100*, we computed the pairwise similarity between over 500 million words in 50 hours using 200 four-core machines. Web004 is of similar scale to the largest reported randomized technique (Ravichandran et al. 2005). On this scale, we compute the exact similarity matrix in a little over two hours whereas Ravichandran et al. (2005) compute an approximation in 570 hours. On average they only find 73%

[3] Available for download at http://www.patrickpantel.com/cgi-bin/Web/Tools/getfile.pl?type=data&id=sse-gold/wikipedia.20071218.trials.tgz.

[4] Note also that we do not allow seed instances nor their variants to appear in an expansion set.

[5] To avoid biasing our Wikipedia corpus with the test sets, Wikipedia "*List of*" pages were omitted from our statistics as were any page linked to gold standard list members from "List of" pages.

Table 3. Corpora analysis: *R*-precision and *Precision* at various ranks. 95% confidence bounds are all below 0.005†.

CORPORA	R-PREC	PREC@25	PREC@50	PREC@100
Web100	**0.404**	**0.407**	**0.347**	**0.278**
Web020	0.356	0.377	0.319	0.250
Web004	0.264	0.353	0.298	0.239
Wikipedia	0.315	0.372	0.314	0.253

†95% confidence bounds are computed over all trials described in Section 5.2.

of the top-1000 similar terms of a random term whereas we find all of them.

For set expansion, experiments have been run on corpora as large as *Web004* and Wikipedia (Sarmento et al. 2007), a corpora 300 times smaller than our Web crawl. Below, we compare the expansion accuracy of Sarmento et al. (2007) on Wikipedia and our Web crawls.

Figure 2 illustrates the precision and recall tradeoff for our four corpora, with 95% confidence intervals computed over all 20,220 trials described in Section 4.2. Table 3 lists the resulting *R*-precision along with the system precisions at ranks 25, 50, and 100 (see Figure 2 for detailed precision analysis). Why are the *precision* scores so low? Compared with previous work that manually select entity types for expansion, such as *countries* and *companies*, our work is the first to evaluate over a large set of randomly selected entity types. On just the *countries* class, our *R*-Precision was *0.816* using *Web100*.

The following sections analyze the effects of various expansion variables: corpus size, corpus quality, seed size, and seed quality.

6.2.1 Corpus Size and Corpus Quality Effect

Not surprisingly, corpus size and quality have a significant impact on expansion performance. Figure 2 and Table 3 quantify this expectation. On our Web crawl corpora, we observe that the full 200+ billion token crawl (*Web100*) has an average *R*-precision 13% higher than 1/5th of the crawl (*Web020*) and 53% higher than 1/25th of the crawl. Figure 2 also illustrates that throughout the full precision/recall curve, *Web100* significantly outperforms *Web020*, which in turn significantly outperforms *Web004*.

The higher text quality *Wikipedia* corpus, which consists of roughly 60 times fewer tokens than

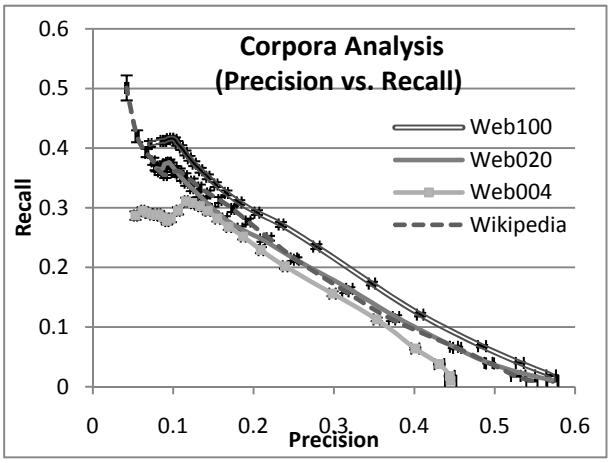


Figure 2. Corpus size and quality improve performance.

Web020, performs nearly as well as *Web020* (see Figure 2). We omitted statistics from Wikipedia "*List of*" pages in order to not bias our evaluation to the test set described in Section 5.1. Inspection of the precision vs. rank graph (omitted for lack of space) revealed that from rank 1 thru 550, *Wikipedia* had the same precision as *Web020*. From rank 550 to 1000, however, *Wikipedia*'s precision dropped off significantly compared with *Web020*, accounting for the fact that the Web corpus contains a higher recall of gold standard instances. The *R*-precision reported in Table 3 shows that this precision drop-off results in a significantly lower *R*-precision for *Wikipedia* compared with *Web020*.

6.2.2 The Effect of Seed Selection

Intuitively, some seeds are better than others. We study the impact of seed selection effect by inspecting the system performance for several randomly selected seed sets of fixed size and we find that ***seed set composition greatly affects performance***. Figure 3 illustrates the precision vs. recall tradeoff on our best performing corpus *Web100* for 30 random seed sets of size 10 for each of our 50 gold standard sets (i.e., 1500 trials were tested.) Each of the trials performed better than the average system performance (the double-lined curve lowest in Figure 3). Distinguishing between the various data series is not important, however important to notice is the very large gap between the precision/recall curves of the best and worst performing random seed sets. On average, the best performing seed sets had 42% higher precision and 39% higher recall than the worst performing seed set. Similar

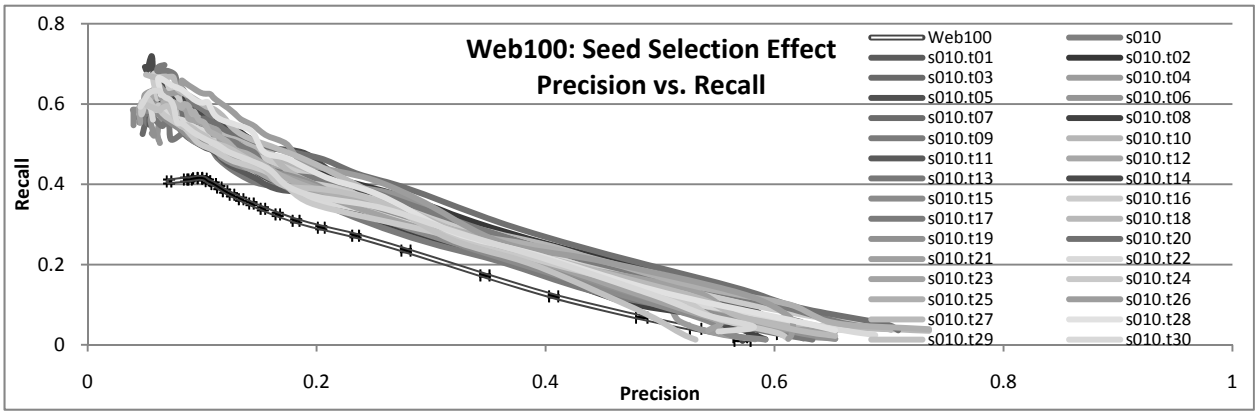


Figure 3. Seed set composition greatly affects system performance (with 30 different seed samples of size 10).

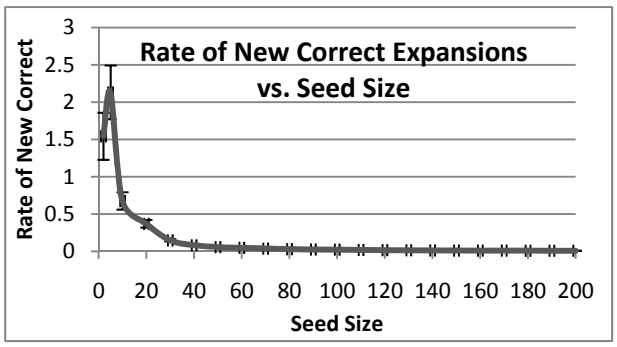


Figure 4. Few new instances are discovered with more than 5-20 seeds on *Web100* (with 95% confidence).

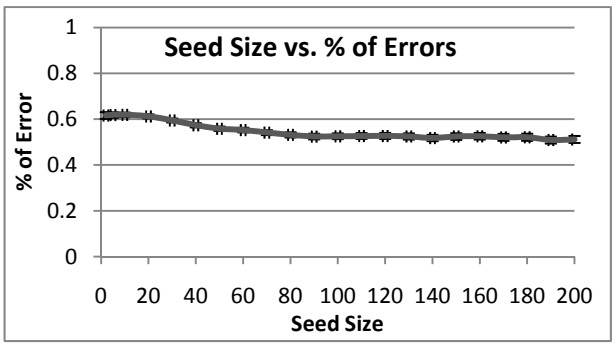


Figure 5. Percentage of errors does not increase as seed size increases on *Web100* (with 95% confidence).

curves were observed for inspected seed sets of size 5, 20, 30, and 40.

Although outside of the scope of this paper, we are currently investigating ways to automatically detect which seed elements are better than others in order to reduce the impact of seed selection effect.

6.2.3 The Effect of Seed Size

Here we aim to confirm, with a large empirical study, the anecdotal claims in (Paşca and Durme 2008) that few seeds are necessary. We found that a) very small seed sets of size 1 or 2 are not sufficient for representing the intended entity set; b) 5-20 seeds yield on average best performance; and c) surprisingly, ***increasing the seed set size beyond 20 or 30 on average does not find any new correct instances***.

We inspected the effect of seed size on *R*-precision over the four corpora. Each seed size curve is computed by averaging the system performance over the 30 random trials of all 50 sets. For each corpus, *R*-precision increased sharply from seed size 1 to 10 and the curve flattened out for seed sizes larger than 20 (figure omitted for lack of space). Error analysis on the *Web100* corpus shows that once our model has seen 10-20 seeds, the distributional similarity model seems to have enough statistics to discover as many new correct instances as it could ever find. Some entities could never be found by the distributional similarity model since they either do not occur or infrequently occur in the corpus or they occur in contexts that vary a great deal from other set elements. Figure 4 illustrates this behavior by plotting for each seed set size the rate of increase in discovery of *new* correct instances (i.e., not found in smaller seed set sizes).

We see that most gold standard instances are discovered with the first 5-10 seeds. After the 30th seed is introduced, no new correct instances are found. An important finding is that the ***error rate does not increase with increased seed set size*** (see Figure 5). This study shows that only few seeds (10-20) yield best performance and that adding more seeds beyond this does not on average affect performance in a positive or negative way.

7 Conclusion

We proposed a highly scalable term similarity algorithm, implemented in the MapReduce framework, and deployed over a 200 billion word crawl of the Web. The pairwise similarity between 500 million terms was computed in 50 hours using 200 quad-core nodes. We evaluated the impact of the large similarity matrix on a set expansion task and found that the Web similarity matrix gave a large performance boost over a state-of-the-art expansion algorithm using Wikipedia. Finally, we release to the community a testbed for experimentally analyzing automatic set expansion, which includes a large collection of nearly random entity sets extracted from Wikipedia and over 22,000 randomly sampled seed expansion trials.

References

Abney, S. Parsing by Chunks. In: Robert Berwick, Steven Abney and Carol Tenny (eds.), *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht. 1991.

Agirre, E.; Alfonseca, E.; Hall, K.; Kravalova, J.; Pasca, M.; and Soroa, A.. 2009. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In *Proceedings of NAACL HLT 09*.

Ando, R. K. 2000. Latent semantic space: Iterative scaling improves precision of interdocument similarity measurement. In *Proceedings of SIGIR-00*. pp. 216–223.

Atterer, M. and Schutze, H., 2006. The Effect of Corpus Size when Combining Supervised and Unsupervised Training for Disambiguation. In *Proceedings of ACL-06*.

Banko, M. and Brill, E. 2001. Mitigating the paucity of data problem. In *Proceedings of HLT-2001*. San Diego, CA.

Banko, M.; Cafarella, M.; Soderland, S.; Broadhead, M.; Etzioni, O. 2007. Open Information Extraction from the Web. In *Proceedings of IJCAI*.

Bayardo, R. J.; Ma, Y.; Srikant, R. 2007. Scaling Up All-Pairs Similarity Search. *In Proceedings of WWW-07*. pp. 131-140. Banff, Canada.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Brill, E. 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging. *Computational Linguistics*.

Broder, A. 1997. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*. pp. 21-29.

Bunescu, R. and Mooney, R. 2004 Collective Information Extraction with Relational Markov Networks. In *Proceedings of ACL-04,* pp. 438-445.

Cao, H.; Jiang, D.; Pei, J.; He, Q.; Liao, Z.; Chen, E.; and Li, H. 2008. Context-aware query suggestion by mining click-through and session data. In *Proceedings of KDD-08*. pp. 875–883.

Chang, W.; Pantel, P.; Popescu, A.-M.; and Gabrilovich, E. 2009. Towards intent-driven bidterm suggestion. In *Proceedings of WWW-09 (Short Paper)*, Madrid, Spain.

Church, K. and Hanks, P. 1989. Word association norms, mutual information, and lexicography. In *Proceedings of ACL89*. pp. 76–83.

Dean, J. and Ghemawat, S. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107-113.

Deerwester, S. C.; Dumais, S. T.; Landauer, T. K.; Furnas, G. W.; and Harshman, R. A. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Downey, D.; Broadhead, M; Etzioni, O. 2007. Locating Complex Named Entities in Web Text. In *Proceedings of IJCAI-07.*

Elsayed, T.; Lin, J.; Oard, D. 2008. Pairwise Document Similarity in Large Collections with MapReduce. In *Proceedings of ACL-08: HLT, Short Papers (Companion Volume)*. pp. 265–268. Columbus, OH.

Erk, K. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of ACL-07*. pp. 216–223. Prague, Czech Republic.

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

Etzioni, O.; Cafarella, M.; Downey. D.; Popescu, A.; Shaked, T; Soderland, S.; Weld, D.; Yates, A. 2005. Unsupervised named-entity extraction from the Web: An Experimental Study. In *Artificial Intelligence,* 165(1):91-134.

Gorman, J. and Curran, J. R. 2006. Scaling distributional similarity to large corpora. In Proceedings of ACL-06. pp. 361-368.

Harris, Z. 1985. Distributional Structure. In: Katz, J. J. (ed.), The Philosophy of Linguistics. New York: Oxford University Press. pp. 26-47.

Hindle, D. 1990. Noun classification from predicate-argument structures. In Proceedings of ACL-90. pp. 268–275. Pittsburgh, PA.

Hofmann, T. 1999. Probabilistic Latent Semantic Indexing. In *Proceedings of SIGIR-99*. pp. 50–57, Berkeley, California.

Kanerva, P. 1993. Sparse distributed memory and related models. pp. 50-76.

Lapata, M. and Keller, F., 2005. Web-based Models for Natural Language Processing, In ACM Transactions on Speech and Language Processing (TSLP), 2(1).

Lee, Lillian. 1999. Measures of Distributional Similarity. In Proceedings of ACL-93. pp. 25-32. College Park, MD.

Lin, D. 1998. Automatic retrieval and clustering of similar words. In Proceedings of COLING/ACL-98. pp. 768–774. Montreal, Canada.

Lund, K., and Burgess, C. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, 28(2):203–208.

McCallum, A. and Li, W. Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Enhanced Lexicons. In *Proceedings of CoNLL-03*.

McQueen, J. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematics, Statistics and Probability*, 1:281–298.

Paşca, M. 2007a. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of CIKM-07*. pp. 683-690.

Paşca, M. 2007b. Organizing and Searching the World Wide Web of Facts – Step Two: Harnessing the Wisdom of the Crowds. In *Proceedings of WWW-07*.

Paşca, M. and Durme, B.J. 2008. Weakly-supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs. *In Proceedings of ACL-08*.

Paşca, M.; Lin, D.; Bigham, J.; Lifchits, A.; Jain, A. 2006. Names and Similarities on the Web: Fast Extraction in the Fast Lane. In *Proceedings of ACL-2006*. pp. 113-120.

Pantel, P. and Lin, D. 2002. Discovering Word Senses from Text. In *Proceedings of KDD-02*. pp. 613-619. Edmonton, Canada.

Pantel, P.; Ravichandran, D; Hovy, E.H. 2004. Towards terascale knowledge acquisition. *In proceedings of COLING-04*. pp 771-777.

Ravichandran, D.; Pantel, P.; and Hovy, E. 2005. Randomized algorithms and NLP: Using locality sensitive hash function for high speed noun clustering. In *Proceedings of ACL-05*. pp. 622-629.

Riloff, E. and Jones, R. 1999 Learning Dictionaries for Information Extraction by Multi-Level Boostrapping. In *Proceedings of AAAI/IAAAI-99*.

Riloff, E. and Shepherd, J. 1997. A corpus-based approach for building semantic lexicons. In *Proceedings of EMNLP-97.*

Rychlý, P. and Kilgarriff, A. 2007. An efficient algorithm for building a distributional thesaurus (and other Sketch Engine developments). In *Proceedings of ACL-07*, demo sessions. Prague, Czech Republic.

Sarawagi, S. and Kirpal, A. 2004. Efficient set joins on similarity predicates. In *Proceedings of SIGMOD '04*. pp. 74 –754. New York, NY.

Sarmento, L.; Jijkuon, V.; de Rijke, M.; and Oliveira, E. 2007. "More like these": growing entity classes from seeds. In *Proceedings of CIKM-07*. pp. 959-962. Lisbon, Portugal.

Turney, P. D., and Littman, M. L. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems, 21(4)*.

Wang, R.C. and Cohen, W.W. 2008. Iterative Set Expansion of Named Entities using the Web. In *Proceedings of ICDM 2008*. Pisa, Italy.

Wang. R.C. and Cohen, W.W. 2007 Language-Independent Set Expansion of Named Entities Using the Web. In *Proceedings of ICDM-07*.

Yuret, D., and Yatbaz, M. A. 2009. The noisy channel model for unsupervised word sense disambiguation. *Computational Linguistics*. Under review.

Toward Completeness in Concept Extraction and Classification

Eduard Hovy and Zornitsa Kozareva
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
hovy@isi.edu, zkozareva@gmail.com

Ellen Riloff
School of Computing
University of Utah
Salt Lake City, UT 84112
riloff@cs.utah.edu

Abstract

Many algorithms extract terms from text together with some kind of taxonomic classification (is-a) link. However, the general approaches used today, and specifically the methods of evaluating results, exhibit serious shortcomings. Harvesting without focusing on a specific conceptual area may deliver large numbers of terms, but they are scattered over an immense concept space, making Recall judgments impossible. Regarding Precision, simply judging the correctness of terms and their individual classification links may provide high scores, but this doesn't help with the eventual assembly of terms into a single coherent taxonomy. Furthermore, since there is no correct and complete gold standard to measure against, most work invents some ad hoc evaluation measure. We present an algorithm that is more precise and complete than previous ones for identifying from web text just those concepts 'below' a given seed term. Comparing the results to WordNet, we find that the algorithm misses terms, but also that it learns many new terms not in WordNet, and that it classifies them in ways acceptable to humans but different from WordNet.

1 Collecting Information with Care

Over the past few years, many algorithms have been published on automatically harvesting terms and their conceptual types from the web and/or other large corpora (Etzioni et al., 2005; Pasca, 2007; Banko et al., 2007; Yi and Niblack, 2005; Snow et al., 2005). But several basic problems limit the eventual utility of the results.

First, there is no standard collection of facts against which results can be measured. As we show in this paper, WordNet (Fellbaum, 1998), the most obvious contender because of its size and popularity, is deficient in various ways: it is neither complete nor is its taxonomic structure inarguably perfect. As a result, alternative ad hoc measures are invented that are not comparable. Second, simply harvesting facts about an entity without regard to its actual subsequent organization inflates Recall and Precision evaluation scores: while it is correct that a *jaguar* is a *animal, mammal, toy, sports-team, car-make*, and *operating-system*, this information doesn't help to create a taxonomy that, for example, places *mammal* and *animal* closer to one another than to some of the others. ((Snow et al., 2005) is an exception to this.) As a result, this work may give a misleading sense of progress. Third, entities are of different formal types, and their taxonomic treatment is consequently different: some are at the level of instances (e.g., *Michelangelo was a painter*) and some at the level of concepts (e.g., *a painter is a human*).

The goal of our research is to learn terms for entities (objects) and their taxonomic organization simultaneously, from text. Our method is to use a single surface-level pattern with several open positions. Filling them in different ways harvests different kinds of information, and/or confirms this information. We evaluate in two ways: against WordNet, since that is a commonly available and popular resource, and also by asking humans to judge the results since WordNet is neither complete nor exhaustively taxonomized.

In this paper, we describe experiments with two rich and common portions of an entity taxonomy: Animals and People. The claim of this paper is: *It is possible to learn terms automatically to populate a targeted portion of a taxonomy (such as below An-*

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 948–957,
Singapore, 6-7 August 2009.

imals or People) both at high precision compared to WordNet and including additional correct ones as well. We would like to also report on Recall relative to WordNet, but given the problems described in Section 4, this turns out to be much harder than would seem.

First, we need to define some basic terminology:
term: An English word (for our current purposes, a noun or a proper name).
seed term: A word we use to initiate the algorithm.
concept: An item in the classification taxonomy we are building. A concept may correspond to several terms (singular form, plural form, the term's synonyms, etc.).
root concept: A concept at a fairly general (high) level in the taxonomy, to which many others are eventually learned to be subtypes/instances of.
basic-level concept: A concept at the 'basic level', corresponding approximately to the Basic Level categories defined in Prototype Theory in Psychology (Rosch, 1978). For our purposes, a concept corresponding to the (proto)typical level of generality of its type; that is, a *dog*, not a *mammal* or a *dachshund*; a *singer*, not a *human* or an *opera diva*.
instance: An item in the classification taxonomy that is more specific than a concept; only one example of the instance exists in 'the real world' at any time. For example, *Michelangelo* is an instance, as well as *Mazda Miata with license plate 3HCY687*, while *Mazda Miata* is not.
classification link: We use a single relation, that, depending on its arguments, is either *is a type of* (when both arguments are concepts), or *is an instance of* or *is an example of* (when the first argument is an instance/example of the second).

Section 2 describes our method for harvesting; Section 3 discusses related work; and Section 4 describes the experiments and the results.

2 Term and Relation Extraction using the Doubly-Anchored Pattern

Our goal is to develop a technique that automatically 'fills in' the concept space in the taxonomy below any root concept, by harvesting terms through repeated web queries. We perform this in two alternating stages.

Stage 1: Basic-level/Instance concept collection: We use the Doubly-Anchored Pattern DAP developed in (Kozareva et al., 2008):

DAP: *[SeedTerm1] such as [SeedTerm2] and <X>*

which learns a list of basic-level concepts or instances (depending on whether SeedTerm2 expresses a basic-level concept or an instance).[1] DAP is very reliable because it is instantiated with examples at both 'ends' of the space to be filled (the higher-level (root) concept SeedTerm1 and a basic-level term or instance (SeedTerm2)), which mutually disambiguate each other. For example, "presidents" for SeedTerm1 can refer to the leader of a country, corporation, or university, and "Ford" for SeedTerm2 can refer to a car company, an automobile pioneer, or a U.S. president. But when the two terms co-occur in a text that matches the pattern *"Presidents such as Ford and <X>"*, the text will almost certainly refer to country presidents.

The first stage involves a series of repeated replacements of SeedTerm2 by newly-learned terms in order to generate even more seed terms. That is, each new basic-level concept or instance is rotated into the pattern (becoming a new SeedTerm2) in a bootstrapping cycle that Kozareva et al. called *reckless bootstrapping*. This procedure is implemented as exhaustive breadth-first search, and iterates until no new terms are harvested. The harvested terms are incorporated in a *Hyponym Pattern Linkage Graph (HPLG)* $G = (V, E)$, where each vertex $v \in V$ is a candidate term and each edge $(u, v) \in E$ indicates that term v was generated by term u. A term u is ranked by *Out-Degree(u)* = $\frac{\sum_{\forall (u,v) \in E} w(u,v)}{|V|-1}$, which represents the weighted sum of u's outgoing edges normalized by the total number of other nodes in the graph. Intuitively, a term ranks highly if it is frequently discovering many different terms during the reckless bootstrapping cycle. This method is very productive, harvesting a constant stream of new terms for basic-level concepts or instances when the taxonomy below the initial root concept SeedTerm1 is extensive (such as for Animals or People).

[1] Strictly speaking, our lowest-level concepts can be instances, basic-level concepts, or concepts below the basic level (e.g., *dachsund*). But for the sake of simplicity we will refer to our lowest-level terms as basic-level concepts and instances.

Stage 2: Intermediate level concept collection: Going beyond (Kozareva et al., 2008), we next apply the Doubly-Anchored Pattern in the 'backward' direction (DAP^{-1}), for any two seed terms representing basic-level concepts or instances:

DAP^{-1}: *<X> such as [SeedTerm1] and [SeedTerm2]*

which harvests a set of concepts, most of them intermediate between the basic level or instance and the initial higher-level seed.

This second stage (DAP^{-1}) has not yet been described in the literature. It proceeds analogously. For pairs of basic-level concepts or instances below the root concept that were found during the first stage, we instantiate DAP^{-1} and issue a new web query. For example, if the term "cats" was harvested by DAP in *"Animals such as dogs and <X>"*, then the pair $< dogs, cats >$ forms the new Web query *"<X> such as dogs and cats"*. We extract up to 2 consecutive nouns from the *<X>* position.

This procedure yields a large number of discovered concepts, but they cannot all be used for further bootstrapping. In addition to practical limitations (such as limits on web querying), many of them are too general–more general than the initial root concept–and could derail the bootstrapping process by introducing terms that stray every further away from the initial root concept. We therefore rank the harvested terms based on the likelihood that they will be productive if they are expanded in the next cycle. Ranking is based on two criteria: (1) the concept should be prolific (i.e., produce many lower-level concepts) in order to keep the bootstrapping process energized, and (2) the concept should be subordinate to the root concept, so that the process stays within the targeted part of the search space.

To perform ranking, we incorporate both the harvested concepts and the basic-level/instance pairs into a *Hypernym Relation Graph (HRG)*, which we define as a bipartite graph $HRG = (V, E)$ with two types of vertices. One set of vertices represents the concepts *(the category vertices (V_c)*, and a second set of vertices represents the basic-level/instance pairs that produced the concepts (*the member pair vertices (V_{mp})*). We create an edge $e(u, v) \in E$ between $u \in V_c$ and $v \in V_{mp}$ when the concept represented by u was harvested by the basic-level/instance pair represented by v, with the weight of the edge defined as the number of times that the lower pair found the concept on the web.

We use the Hypernym Relation Graph to rank the intermediate concepts based on each node's *In-Degree*, which is the sum of the weights on the node's incoming edges. Formally, *In-Degree(u)* = $\sum_{\forall(u,v)\in E} w(u, v)$. Intuitively, a concept will be ranked highly if it was harvested by many different combinations of basic-level/instance terms.

However, this scoring function does not determine whether a concept is more or less general than the initial root concept. For example, when harvesting animal categories, the system may learn the word *"species"*, which is a very common term associated with animals, but also applies to non-animals such as plants. To prevent the inclusion of overgeneral terms and constrain the search to remain 'below' the root concept, we apply a *Concept Positioning Test (CPT)*: We issue the following two web queries:

(a) *Concept such as RootConcept and <X>*
(b) *RootConcept such as Concept and <X>*

If (b) returns more web hits than (a), then the concept passes the test, otherwise it fails. The first (most highly ranked) concept that passes CPT becomes the new seed concept for the next bootstrapping cycle. In principle, we could use all the concepts that pass the CPT for bootstrapping[2]. However, for practical reasons (primarily limitations on web querying), we run the algorithm for 10 iterations.

3 Related Work

Many algorithms have been developed to automatically acquire semantic class members using a variety of techniques, including co-occurrence statistics (Riloff and Shepherd, 1997; Roark and Charniak, 1998), syntactic dependencies (Pantel and Ravichandran, 2004), and lexico-syntactic patterns (Riloff and Jones, 1999; Fleischman and Hovy, 2002; Thelen and Riloff, 2002).

The work most closely related to ours is that of (Hearst, 1992) who introduced the idea of applying *hyponym patterns* to text, which explicitly identify a hyponym relation between two terms (e.g.,

[2]The number of ranked concepts that pass CPT changes in each iteration. Also, the wildcard * is important for counts, as can be verified with a quick experiment using Google.

"such authors as <X>"). In recent years, several researchers have followed up on this idea using the web as a corpus. (Pasca, 2004) applies lexico-syntactic hyponym patterns to the Web and use the contexts around them for learning. KnowItAll (Etzioni et al., 2005) applies the hyponym patterns to extract instances from the Web and ranks them by relevance using mutual information. (Kozareva et al., 2008) introduced a bootstrapping scheme using the doubly-anchored pattern (DAP) that is guided through graph ranking. This approach reported a significant improvement from 5% to 18% over approaches using singly-anchored patterns like those of (Pasca, 2004) and (Etzioni et al., 2005).

(Snow et al., 2005) describe a dependency path based approach that generates a large number of weak hypernym patterns using pairs of noun phrases present in WordNet. They build a classifier using the different hypernym patterns and find among the highest precision patterns those of (Hearst, 1992). Snow et al. report performance of 85% precision at 10% recall and 25% precision at 30% recall for 5300 hand-tagged noun phrase pairs. (McNamee et al., 2008) use the technique of (Snow et al., 2005) to harvest the hypernyms of the proper names. The average precision on 75 automatically detected categories is 53%. The discovered hypernyms were intergrated in a Question Answering system which showed an improvement of 9% when evaluated on a TREC Question Answering data set.

Recently, (Ritter et al., 2009) reported hypernym learning using (Hearst, 1992) patterns and manually tagged common and proper nouns. All hypernym candidates matching the pattern are acquired, and the candidate terms are ranked by mutual information. However, they evaluate the performance of their hypernym algorithm by considering only the top 5 hypernyms given a basic-level concept or instance. They report 100% precision at 18% recall, and 66% precision at 72% recall, considering only the top-5 list. Necessarily, using all the results returned will result in lower precision scores. In contrast to their approach, our aim is to first acquire automatically with minimal supervision the basic-level concepts for given root concept. Thus, we almost entirely eliminate the need for humans to provide hyponym seeds. Second, we evaluate the performance of our approach not by measuring the top-ranked 5 hypernyms given a basic-level concept, but considering all harvested hypernyms of the concept.

Unlike (Etzioni et al., 2005), (Pasca, 2007) and (Snow et al., 2005), we learn both instances and concepts simultaneously.

Some researchers have also worked on reorganizing, augmenting, or extending semantic concepts that already exist in manually built resources such as WordNet (Widdows and Dorow, 2002; Snow et al., 2005) or Wikipedia (Ponzetto and Strube, 2007). Work in automated ontology construction has created lexical hierarchies (Caraballo, 1999; Cimiano and Volker, 2005; Mann, 2002), and learned semantic relations such as meronymy (Berland and Charniak, 1999; Girju et al., 2003).

4 Evaluation

The root concepts discussed in this paper are Animals and People, because they head large taxonomic structures that are well-represented in WordNet. Throughout these experiments, we used as the initial SeedTerm2 *lions* for Animals and *Madonna* for People (by specifically choosing a proper name for People we force harvesting down to the level of individual instances). To collect data, we submitted the DAP patterns as web queries to Google, retrieved the top 1000 web snippets per query, and kept only the unique ones. In total, we collected 1.1 GB of snippets for Animals and 1.5 GB for People. The algorithm was allowed to run for 10 iterations.

The algorithm learns a staggering variety of terms that is much more diverse than we had anticipated. In addition to many basic-level concepts or instances, such as *dog* and *Madonna* respectively, and many intermediate concepts, such as *mammals*, *pets*, and *predators*, it also harvested categories that clearly seemed useful, such as *laboratory animals*, *forest dwellers*, and *endangered species*. Many other harvested terms were more difficult to judge, including *bait*, *allergens*, *seafood*, *vectors*, *protein*, and *pests*. While these terms have an obvious relationship to Animals, we have to determine whether they are legitimate and valuable subconcepts of Animals.

A second issue involves relative terms that are hard to define in an absolute sense, such as *native animals* and *large mammals*.

A complete evaluation should answer the following three questions:

- Precision: What is the correctness of the harvested concepts? (How many of them are simply wrong, given the root concept?)
- Recall: What is the coverage of the harvested concepts? (How many are missing, below a given root concept?)
- How correct is the taxonomic structure learned?

Given the number and variety of terms obtained, we initially decided that an automatic evaluation against existing resources (such as WordNet or something similar) would be inadequate because they do not contain many of our harvested terms, even though many of these terms are clearly sensible and potentially valuable. Indeed, the whole point of our work is to learn concepts and taxonomies that go above and beyond what is currently available.

However, it is necessary to compare with *some*thing, and it is important not to skirt the issue by conducting evaluations that measure subsets of results, or that perhaps may mislead. We therefore decided to compare our results against WordNet *and* to have human annotators judge as many results as we could afford (to obtain a measure of Precision and the legitimate extensions beyond WordNet).

Unfortunately, it proved impossible to measure Recall against WordNet, because this requires ascertaining the number of synsets in WordNet between the root and its basic-level categories. This requires human judgment, which we could not afford. We plan to address this question in future work. Also, assessing the correctness of the learned taxonomy structure requires the manual assessment of each classification link proposed by the system that is not already in WordNet, a task also beyond our budget to complete in full. Some results—for just basic-level terms and intermediate concepts, but not among intermediate-level concepts–are shown in Section 4.3.

We provide Precision scores using the following measures, where *terms* refers to the harvested terms:

$$Pr_{WN} = \frac{\#terms\ found\ in\ WordNet}{\#terms\ harvested\ by\ system}$$

$$Pr_{H} = \frac{\#terms\ judged\ correct\ by\ human}{\#terms\ harvested\ by\ system}$$

$$NotInWN = \#terms\ judged\ correct\ by\ human\ but\ not\ in\ WordNet$$

We conducted three sets of experiments. **Experiment 1** evaluates the results of using DAP to learn basic-level concepts for Animals and instances for People. **Experiment 2** evaluates the results of using DAP^{-1} to harvest intermediate concepts between each root concept and its basic-level concepts or instances. **Experiment 3** evaluates the taxonomy structure that is produced via the links between the instances and intermediate concepts.

4.1 Experiment 1: Basic-Level Concepts and Instances

In this section we discuss the results of harvesting the basic-level Animal concepts and People instances. The bootstrapping algorithm ranks the harvested terms by their *Out-Degree* score and considers as correct only those with *Out-Degree* > 0. In ten iterations, the bootstrapping algorithm produced 913 Animal basic-level concepts and $1,344$ People instances that passed this *Out-Degree* criterion.

4.1.1 Human Evaluation

The harvested terms were labeled by human judges as either correct or incorrect with respect to the root concept. Table 1 shows the Precision of the top-ranked N terms, with N shown in increments of 100. Overall, the Animal terms yielded 71% (649/913) Precision and the People terms yielded 95% Precision (1,271/1,344). Figure 1 shows that higher-ranked Animal terms are more accurate than lower-ranked terms, which indicates that the scoring function did its job. For People terms, accuracy was very high throughout the ranked list. Overall, these results show that the bootstrapping algorithm generates a large number of correct instances of high quality.

4.1.2 WordNet Evaluation

Table 1 shows a comparison of the harvested terms against the terms present in WordNet. Note that the Precision measured against WordNet (Pr_{WN}) for People is dramatically different from the Precision based on human judgments (Pr_H). This can be explained by looking at the *NotInWN* column, which shows that 48 correct Animal terms

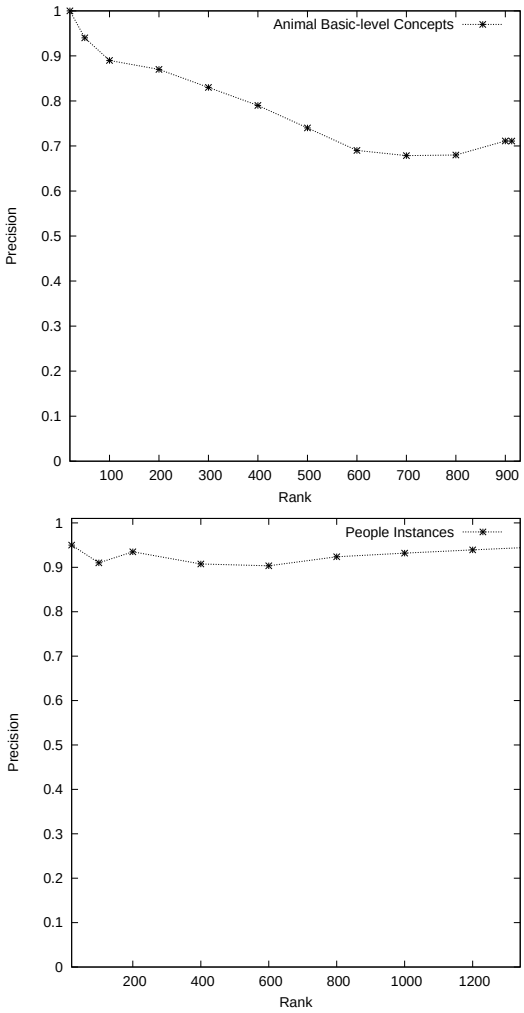


Figure 1: *Ranked Basic-Concepts and Instances.*

and 986 correct People instances are not present in WordNet (primarily, for people, because WordNet contains relatively few proper names). These results show that there is substantial room for improvement in WordNet's coverage of these categories. For Animals, the precision measured against WordNet is actually higher than the precision measured by human judges, which may indicate that the judges failed to recognize some correct animal terms.

	Pr_{WN}	Pr_H	$NotInWN$
Animal	.79	.71	48
People	.23	.95	986

Table 1: *Instance Evaluation.*

4.1.3 Evaluation against Prior Work

To assess how well our algorithm compares with previous semantic class learning methods, we compared our results to those of (Kozareva et al., 2008). Our work was inspired by that approach–in fact, we use that previous algorithm as the first step of our bootstrapping process. The novelty of our approach is the insertion of an additional bootstrapping stage that iteratively learns new intermediate concepts using DAP^{-1} and the Concept Positioning Test, followed by the subsequent use of the newly learned intermediate concepts in DAP to expand the search space beyond the original root concept. This leads to the discovery of additional basic-level terms or instances, which are then recycled in turn to discover new intermediate concepts, and so on.

Consequently, we can compare the results produced by the first iteration of our algorithm (before intermediate concepts are learned) to those of (Kozareva et al., 2008) for the Animal and People categories, and then compare again after 10 bootstrapping iterations of intermediate concept learning. Figure 2 shows the number of harvested concepts for Animals and People after each bootstrapping iteration. Bootstrapping with intermediate concepts produces nearly 5 times as many basic-level concepts and instances than (Kozareva et al., 2008) obtain, while maintaining similar levels of precision.

The intermediate concepts help so much because they steer the learning process into new (yet still correct) regions of the search space after each iteration. For instance, in the first iteration, the pattern *"animals such as lions and *"* harvests about 350 basic-level concepts, but only animals that are mentioned in conjunction with lions are learned. Of these, animals typically quite different from lions, such as grass-eating kudu, are often not discovered.

However, in the second iteration, the intermediate concept Herbivore is chosen for expansion. The pattern *"herbivore such as antelope and *"* discovers many additional animals, including *kudu*, that co-occur with *antelope* but do not co-occur with *lions*.

Table 2 shows examples of the 10 top-ranked basic-level concepts and instances that were learned for 3 randomly-selected intermediate Animal and People concepts (*IConcepts*) that were acquired during bootstrapping. In the next section, we present an

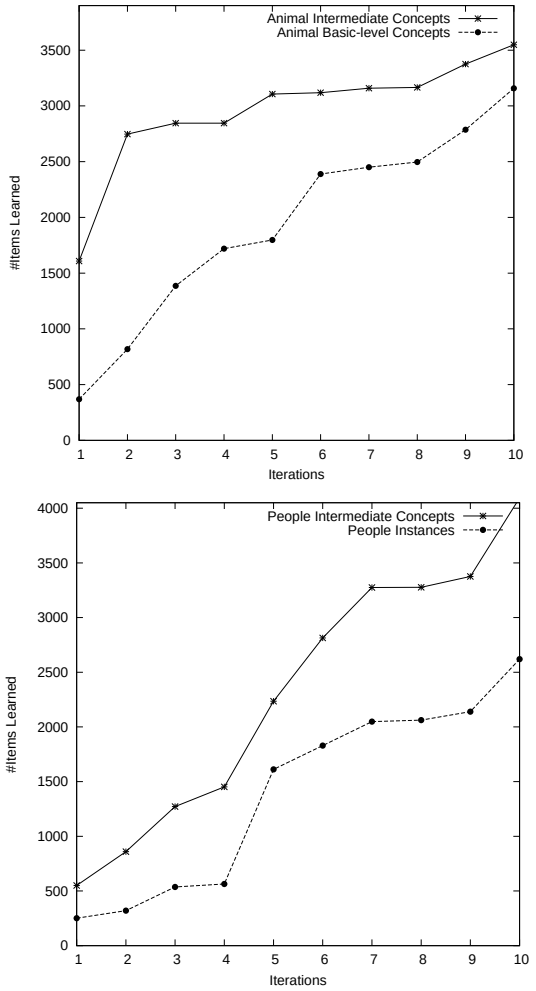


Figure 2: *Learning Curves.*

evaluation of the intermediate concept terms.

4.2 Experiment 2: Intermediate Concepts

In this section we discuss the results of harvesting the intermediate-level concepts. Given the variety of the harvested results, manual judgment of correctness required an in-depth human annotation study. We also compare our harvested results against the concept terms in WordNet.

4.2.1 Human Evaluation

We hired 4 annotators (undergraduates at a different institution) to judge the correctness of the intermediate concepts. We created detailed annotation guidelines that define 14 annotation labels for each of the Animal and People classes, as shown in Table 3. The labels are clustered into 4 major types: *Correct*, *Borderline*, *BasicConcept, and NotConcept*. The details of our annotation guidelines, the reasons for the intermediate labels, and the annotation study can be found in (Kozareva et al., 2009).

PEOPLE	
IConcept	*Instances*
Dictators:	Adolf Hitler, Joseph Stalin, Benito Mussolini, Lenin, Fidel Castro, Idi Amin, Slobodan Milosevic, Hugo Chavez, Mao Zedong, Saddam Hussein
Celebrities:	Madonna, Paris Hilton, Angelina Jolie, Britney , Spears, Tom Cruise, Cameron Diaz, Bono, Oprah Winfrey, Jennifer Aniston, Kate Moss
Writers:	William Shakespeare, James Joyce, Charles Dickens, Leo Tolstoy, Goethe, Ralph Waldo Emerson, Daniel Defoe, Jane Austen, Ernest Hemingway, Franz Kafka

ANIMAL	
IConcept	*Basic-level Terms*
Crustacean:	shrimp, crabs, prawns, lobsters, crayfish, mysids, decapods, marron, ostracods, yabbies
Primates:	baboons, monkeys, chimpanzees, apes, marmosets, chimps, orangutans, gibbons, tamarins, bonobos
Mammal:	mice, whales, seals, dolphins, rats, deer, rabbits, dogs, elephants, squirrels

Table 2: *Learned People and Animals Terms.*

ANIMAL

TYPE	LABEL	EXAMPLES
Correct	GeneticAnimal	*reptile,mammal*
	BehavioralByFeeding	*predator, grazer*
	BehaviorByHabitat	*saltwater mammal*
	BehaviorSocialIndiv	*herding animal*
	BehaviorSocialGroup	*herd, pack*
	MorphologicalType	*cloven-hoofed animal*
	RoleOrFunction	*pet, parasite*
Borderline	NonRealAnimal	*dragons*
	EvaluativeTerm	*varmint, fox*
	OtherAnimal	*critter, fossil*
BasicConcept	BasicAnimal	*dog, hummingbird*
NotConcept	GeneralTerm	*model, catalyst*
	NotAnimal	*topic, favorite*
	GarbageTerm	*brates, mals*

PEOPLE

TYPE	LABEL	EXAMPLES
Correct	GeneticPerson	*Caucasian, Saxon*
	NonTransientEventRole	*stutterer, gourmand*
	TransientEventRole	*passenger, visitor*
	PersonState	*dwarf, schizophrenic*
	FamilyRelation	*aunt, mother*
	SocialRole	*fugitive, hero*
	NationOrTribe	*Bulgarian, Zulu*
	ReligiousAffiliation	*Catholic, atheist*
Borderline	NonRealPerson	*biblical figures*
	OtherPerson	*colleagues, couples*
BasicConcept	BasicPerson	*child, woman*
	RealPerson	*Barack Obama*
NotConcept	GeneralTerm	*image, figure*
	NotPerson	*books, events*

Table 3: *Intermediate Concept Annotation Labels*

We measured pairwise inter-annotator agreement across the four labels using the Fleiss kappa (Fleiss, 1971). The κ scores ranged from 0.61–0.71 for Animals (average κ=0.66) and from 0.51–0.70 for People (average κ=0.60). These agreement scores seemed good enough to warrant using these human judgments to estimate the accuracy of the algorithm.

The bootstrapping algorithm harvested $3,549$ Animal and $4,094$ People intermediate concepts in ten iterations. After *In-Degree* ranking was applied,

we chose a random sample of intermediate concepts with frequency over 1, which was given to four human judges for annotation. Table 4 summarizes the labels assigned by the four annotators (A_1 – A_4). The top portion of Table 4 shows the results for all the intermediate concepts (437 Animal terms and 296 People terms), and the bottom portion shows the results only for the concepts that passed the Concept Positioning Test (187 Animal terms and 139 People terms). Accuracy is computed in two ways: **Acc1** is the percent of intermediate concepts labeled as *Correct*; **Acc2** is the percent of intermediate concepts labeled as either *Correct* or *Borderline*.

Without the CPT, accuracies range from 53–66% for Animals and 75–85% for People. After applying the CPT, the accuracies increase to 71–84% for animals and 82–94% for people. These results confirm that the Concept Positioning Test is effective at removing many of the undesirable terms. Overall, these results demonstrate that our algorithm produced many high-quality intermediate concepts, with good precision.

Figure 3 shows accuracy curves based on the rankings of the intermediate concepts (based on In-Degree scores). The CPT clearly improves accuracy even among the most highly ranked concepts. For example, the **Acc1** curves for animals show that nearly 90% of the top 100 intermediate concepts were correct after applying the CPT, whereas only 70% of the top 100 intermediate concepts were correct before. However, the CPT also eliminates many desirable terms. For People, the accuracies are still relatively high even without the CPT, and a much larger set of intermediate concepts is learned.

	Animals				**People**			
	A_1	A_2	A_3	A_4	A_1	A_2	A_3	A_4
Correct	246	243	251	230	239	231	225	221
Borderline	42	26	22	29	12	10	6	4
BasicConcept	2	8	9	2	6	2	9	10
NotConcept	147	160	155	176	39	53	56	61
Acc1	.56	.56	.57	.53	.81	.78	.76	.75
Acc2	.66	.62	.62	.59	.85	.81	.78	.76
	Animals after CPT				**People after CPT**			
	A_1	A_2	A_3	A_4	A_1	A_2	A_3	A_4
Correct	146	133	144	141	126	126	114	116
Borderline	11	15	9	13	6	2	2	0
BasicConcept	2	8	9	2	0	1	7	7
NotConcept	28	31	25	31	7	10	16	16
Acc1	.78	.71	.77	.75	.91	.91	.82	.83
Acc2	.84	.79	.82	.82	.95	.92	.83	.83

Table 4: *Human Intermediate Concept Evaluation.*

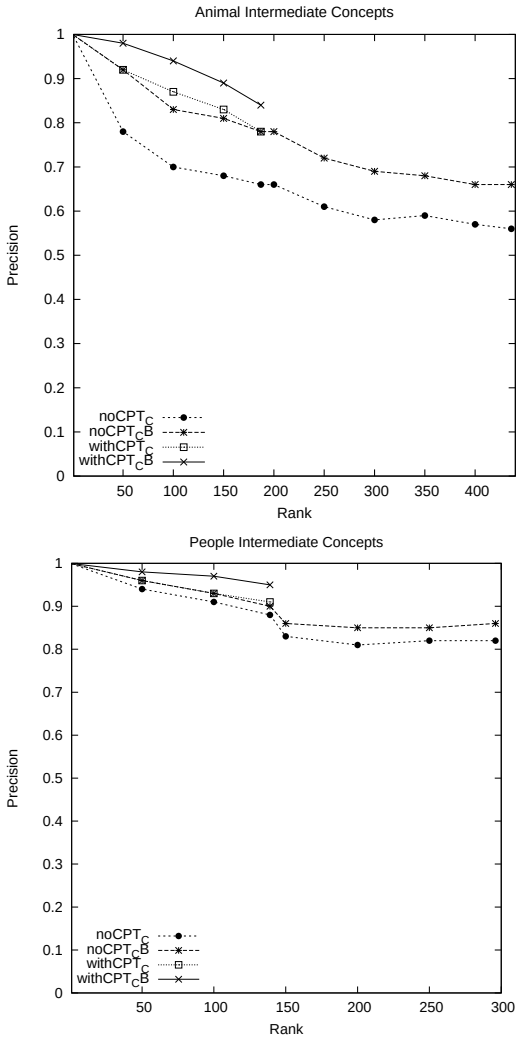


Figure 3: *Intermediate Concept Precision at Rank N.*

4.2.2 WordNet Evaluation

We also compared the intermediate concepts harvested by the algorithm to the contents of WordNet. The results are shown in Table 5. WordNet contains 20% of the Animal concepts and 51% of the People concepts learned by our algorithm, which confirms that many of these concepts were considered to be valuable taxonomic terms by the WordNet developers. However, our human annotators judged 57% of the Animal and 84% of the People concepts to be correct, which suggests that our algorithm generates a substantial number of additional concepts that could be used to enrich taxonomic structure in WordNet.

	Pr_{WN}	Pr_H	NotInWN
Animal	.20 (88/437)	.57 (248/437)	204
People	.51 (152/296)	.85 (251/296)	108

Table 5: *WordNet Intermediate Concept Evaluation.*

4.3 Experiment 3: Taxonomic Links

In this section we evaluate the classification (taxonomy) that is learned by evaluating the links between the intermediate concepts and the basic-level concept/instance terms. That is, when our algorithm claims that *isa(X,Y)*, how often is *X* truly a subconcept of *Y*? For example, *isa(goat, herbivore)* would be correct, but *isa(goat, bird)* would not. Again, since WordNet does not contain all the harvested concepts, we conduct both a manual evaluation and a comparison against WordNet.

4.3.1 Manual and WordNet Evaluations

Creating and evaluating the full taxonomic structure between the root and the basic-level or instance terms is future work. Here we evaluate simply the accuracy of the taxonomic links between basic-level concepts/instances and intermediate concepts as harvested, but not between intermediate concepts. For each pair, we extracted all harvested links and determined whether the same links appear in WordNet. The links were also given to human judges. Table 6 shows the results.

ISA	Pr_{WN}	Pr_H	NotInWN
Animal	.47(912/1940)	.88 (1716/1940)	804
People	.23 (318/908)	.94 (857/908)	539

Table 6: *WordNet Taxonomic Evaluation.*

The results show that WordNet lacks nearly half of the taxonomic relations that were generated by the algorithm: 804 Animal and 539 People links.

5 Conclusion

We describe a novel extension to the DAP approach for discovering basic-level concepts or instances and their superconcepts given an initial root concept. By appropriate filling of different positions in DAP, the algorithm alternates between 'downward' and 'upward' learning. A key resulting benefit is that each new intermediate-level term acquired restarts harvesting in a new region of the concept space, which allows previously unseen concepts to be discovered with each bootstrapping cycle.

We also introduce the *Concept Positioning Test*, which serves to confirm that a harvested concept falls into the desired part of the search space relative to either a superordinate or subordinate concept in the growing taxonomy, before it is selected for further harvesting using the DAP.

These algorithms can augment other term harvesting algorithms recently reported. But in order to compare different algorithms, it is important to compare results to a standard. WordNet is our best candidate at present. But WordNet is incomplete. Our results include a significantly large number of instances of People (which WordNet does not claim to cover), a number comparable to the results of (Etzioni et al., 2005; Pasca, 2007; Ritter et al., 2009). Rather surprisingly, our results also include a large number of basic-level and intermediate concepts for Animals that are not present in WordNet, a category WordNet is actually fairly complete about. These numbers show clearly that it is important to conduct manual evaluation of term harvesting algorithms in addition to comparing to a standard resource.

Acknowledgments

This research was supported in part by grants from the National Science Foundation (NSF grant no. IIS-0429360), and the Department of Homeland Security, ONR Grant numbers N0014-07-1-0152 and N00014-07-1-0149. We are grateful to the annotators at the University of Pittsburgh who helped us evaluate this work: Jay Fischer, David Halpern, Amir Hussain, and Taichi Nakatani.

References

M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O.Etzioni. 2007. Open information extraction from the web. In *Proceedings of International Joint Conference on Artificial Itelligence*, pages 2670–2676.

M. Berland and E. Charniak. 1999. Finding Parts in Very Large Corpora. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics*.

S. Caraballo. 1999. Automatic Acquisition of a Hypernym-Labeled Noun Hierarchy from Text. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 120–126.

P. Cimiano and J. Volker. 2005. Towards large-scale, open-domain and ontology-based named entity classification. In *Proceeding of RANLP-05*, pages 166–172.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence*, 165(1):91–134, June.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. May.

M.B. Fleischman and E.H. Hovy. 2002. Fine grained classification of named entities. In *Proceedings of the COLING conference*, August.

J.L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

R. Girju, A. Badulescu, and D. Moldovan. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. In *HLT-NAACL*.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545.

Z. Kozareva, E. Riloff, and E. Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056. Association for Computational Linguistics.

Z. Kozareva, E. Hovy, and E. Riloff. 2009. Learning and evaluating the content and structure of a term taxonomy. In *AAAI-09 Spring Symposium on Learning by Reading and Learning to Read*.

G. Mann. 2002. Fine-grained proper noun ontologies for question answering. In *COLING-02 on SEMANET*, pages 1–7.

P. McNamee, R. Snow, P. Schone, and J. Mayfield. 2008. Learning named entity hyponyms for question answering. In *Proceedings of the Third International Joint Conference on Natural Language Processing*.

P. Pantel and D. Ravichandran. 2004. Automatically labeling semantic classes. In *HLT-NAACL*, pages 321–328.

M. Pasca. 2004. Acquisition of categorized named entities for web search. In *Proceedings of CIKM*, pages 137–145.

M. Pasca. 2007. Weakly-supervised discovery of named entities using web search queries. In *CIKM*, pages 683–690.

S. Ponzetto and M. Strube. 2007. Deriving a large scale taxonomy from wikipedia. In *Proceedings of the 22nd National COnference on Artificial Intelligence (AAAI-07)*, pages 1440–1447.

E. Riloff and R. Jones. 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*.

E. Riloff and J. Shepherd. 1997. A Corpus-Based Approach for Building Semantic Lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124.

A. Ritter, S. Soderland, and O. Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Proceedings of AAAI-09 Spring Symposium on Learning by Reading and Learning to Read*, pages 88–93.

B. Roark and E. Charniak. 1998. Noun-phrase Co-occurrence Statistics for Semi-automatic Semantic Lexicon Construction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 1110–1116.

E. Rosch, 1978. *Principles of Categorization*, pages 27–48.

R. Snow, D. Jurafsky, and A. Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

M. Thelen and E. Riloff. 2002. A Bootstrapping Method for Learning Semantic Lexicons Using Extraction Pattern Contexts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 214–221.

D. Widdows and B. Dorow. 2002. A graph model for unsupervised lexical acquisition. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7.

J. Yi and W. Niblack. 2005. Sentiment mining in webfountain. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, pages 1073–1083.

Reading to Learn: Constructing Features from Semantic Abstracts

Jacob Eisenstein[*] **James Clarke**[†] **Dan Goldwasser**[†] **Dan Roth**[*†]
[*]Beckman Institute for Advanced Science and Technology, [†]Department of Computer Science
University of Illinois
Urbana, IL 61801
{jacobe,clarkeje,goldwas1,danr}@illinois.edu

Abstract

Machine learning offers a range of tools for training systems from data, but these methods are only as good as the underlying representation. This paper proposes to acquire representations for machine learning by reading text written to accommodate human learning. We propose a novel form of semantic analysis called *reading to learn*, where the goal is to obtain a high-level semantic abstract of multiple documents in a representation that facilitates learning. We obtain this abstract through a generative model that requires no labeled data, instead leveraging repetition across multiple documents. The semantic abstract is converted into a transformed feature space for learning, resulting in improved generalization on a relational learning task.

1 Introduction

Machine learning offers a range of powerful tools for training systems to act in complex environments, but these methods depend on a well-chosen representation for features. For learning to succeed the representation often must be crafted with knowledge about the application domain. This poses a bottleneck, requiring expertise in both machine learning and the application domain. However, domain experts often express their knowledge through text; one direct expression is through text designed to aid human learning. In this paper we exploit text written by domain experts in order to build a more expressive representation for learning. We term this approach *reading to learn*.

The following scenario demonstrates the motivation for reading to learn. Imagine an agent given a task within its world/environment. The agent has no prior knowledge of the task but can perceive the world through low-level sensors. Learning directly from the sensors may be difficult, as interesting tasks typically require a complex combination of sensors. Our goal is to acquire domain knowledge through the semantic analysis of text, so as to produce higher-level relations through combinations of sensors.

As a concrete example consider the problem of learning how to make legal moves in Freecell solitaire. Relevant sensors may indicate if an object is a card or a freecell, whether a card is a certain value, and whether two values are in sequence. Although it is possible to express the rules with a combination of sensors, learning this combination is difficult. Text can facilitate learning by providing relations at the appropriate level of generalization. For example, the sentence: "You can place a card on an empty freecell," suggests not only which sensors are useful together but also how these sensors should be linked. Assuming the sensors are represented as predicates, one possible relation this sentence suggests is: $\texttt{r}(x, y) = \texttt{card}(x) \wedge \texttt{freecell}(y) \wedge \texttt{empty}(y)$. Armed with this new relation the agent's learning task may be simpler. Throughout the paper we refer to low-level sensory input as *sensor* or *predicate*, and to a higher level concept as a *logical formula* or *relation*.

Our approach to semantic analysis does not require a complete semantic representation of the text. We merely wish to acquire a *semantic abstract* of a document or document collection, and use the discovered relations to facilitate data-driven learning. This will allow us to directly evaluate the contribution of the extracted relations for learning.

We develop an approach to recover semantic abstracts that uses minimal supervision: we assume only a very small set of lexical *glosses*, which map from words to sensors. This marks a substantial departure from previous work on semantic parsing, which requires either annotations of the meanings of each individual sentence (Zettlemoyer and Collins, 2005; Liang et al., 2009), or alignments of sentences to grounded representations of the

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 958–967,
Singapore, 6-7 August 2009.

world (Chen and Mooney, 2008). For the purpose of learning, this approach may be inapplicable, as such text is often written at a high level of abstraction that permits no grounded representation.

There are two properties of our setting that make unsupervised learning feasible. First, it is not necessary to extract a semantic representation of each individual sentence, but rather a summary of the semantics of the document collection. Errors in the semantic abstract are not fatal, as long it guides the learning component towards a more useful representation. Second, we can exploit repetition across documents, which should generally express the same underlying meaning. Logical formulae that are well-supported by multiple documents are especially likely to be useful.

The rest of this paper describes our approach for recovering semantic abstracts and outlines how we apply and evaluate this approach on the Freecell domain. The paper contributes the following key ideas: (1) Interpreting abstract "instructional" text, written at a level that does not correspond to concrete sensory inputs in the world, so that no grounded representation is possible, (2) *reading to learn*, a new setting in which extracted semantic representations are evaluated by whether they facilitate learning; (3) *abstractive semantic summarization*, aimed at capturing broad semantic properties of a multi-document dataset, rather than a semantic parse of individual sentences; (4) a novel, minimally-supervised generative model for semantic analysis which leverages both lexical and syntactic properties of text.

2 Approach Overview

We describe our approach to text analysis as *multidocument semantic abstraction*, with the goal of discovering a compact set of logical formulae to explain the text in a document collection. To this end, we develop a novel generative model in which natural language sentences (e.g., "You can always place cards in empty freecells") are stochastically generated from logical formulae (e.g., $\texttt{card}(x) \wedge \texttt{freecell}(y) \wedge \texttt{empty}(y)$). We formally define a generative process that reflects our intuitions about the relationship between formulae and sentences (Section 3), and perform sampling-based inference to recover the formulae most likely to have generated the observed data (Section 4). The top N such formulae can then be added as additional predicates for relational learning.

Our semantic representation consists of conjunctions of *literals*, each of which includes a single *predicate* (e.g., `empty`) and one or more *variables* (e.g., x). Predicates describe atomic semantic concepts, while variables construct networks of relationships between them. While the importance of the predicates is obvious, the variable assignments also exert a crucial influence on the semantics of the conjunction: modifying a single variable in the formula above from $\texttt{empty}(y)$ to $\texttt{empty}(x)$ yields a formula that is trivially false for all groundings (since cards can never be empty).

Thus, our generative model must account for the influence of both predicates and variables on the sentences in the documents. A natural choice is to use the predicates to influence the lexical items, while letting the variables determine the syntactic structure. For example, the formula $\texttt{card}(x) \wedge \texttt{freecell}(y) \wedge \texttt{empty}(y)$ contains three predicates and two variables. The predicates influence the lexical items in a direct way: we expect that sentences generated from this formula will include a member of the gloss set for each predicate – the sentence "Put the cards on the empty freecells" should be more likely than "Columns are constructed by playing cards in alternating colors."

The impact of the variables on the generative process is more subtle. The sharing of the variable y suggests a relationship between the predicates `freecell` and `empty`. This should be realized in the syntactic structure of the sentence. Modeling syntax using a dependency tree, we expect that the glosses for predicates that share terms will appear in compact sub-trees, while predicates that do not share terms should be more distant. One possible surface realization of this logical formula is the sentence, "Put the card on the empty freecell," whose dependency parse is shown in the left tree of Figure 1. The glosses *empty* and *freecell* are immediately adjacent, while *card* is more remote.

We develop two metrics that quantify the compactness of a set of variable assignments with respect to a dependency tree: *excess terms*, and *shared terms*. The number of excess terms in a subtree is the number of unique terms assigned to words in the subtree, minus the maximum arity of any predicate in the subtree. Shared terms arise whenever a node has multiple subtrees which each contain the same variable. We will use the alternative alignments in Figure 1 to provide a more detailed explanation. In each tree, the variables are written in the nodes belonging to the associated lexical items; variables are written over arrows to indicate membership in some node in the subtree.

Excess Terms Alignment A of Figure 1, corresponding to the formula

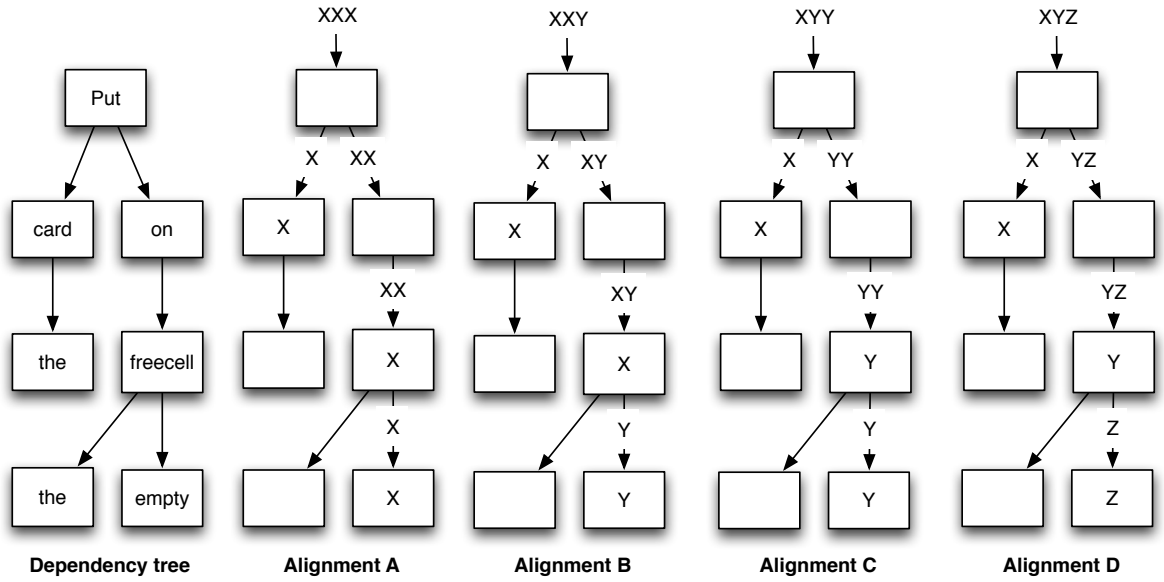


Figure 1: A dependency parse and four different variable assignments. Each literal is aligned to a word (a node in the graph), and the associated variables are written in the box. Variables belonging to descendant nodes are written over the arrows.

$\texttt{card}(x) \wedge \texttt{freecell}(x) \wedge \texttt{empty}(x)$, has zero excess terms in every subtree; there is a total of one variable, and all the predicates are unary. In Alignment B, $\texttt{card}(x) \wedge \texttt{freecell}(x) \wedge \texttt{empty}(y)$, there are excess terms at the root, and in the top two subtrees on the right-hand side. Alignment C contains an excess term at only the root node. Even though it contains the same number of unique variables as Alignment B, it is not penalized as harshly because the alignment of variables better corresponds to the syntactic structure. Alignment D contains the greatest number of excess terms: two at the root of the tree, and one in each of the top two subtrees on the right side.

Shared Terms According to the excess term metric, the best choice is simply to introduce as few variables as possible. For this reason, we also penalize *shared terms* which occur when a node has subtree children that share a variable. In Figure 1, Alignments A and B each contain a shared term at the top node; Alignments C and D contain no shared terms.

Overall, we note that Alignment B is penalized on both metrics, as it contains both excess terms and shared terms; the syntactic structure of the sentence makes such a variable assignment relatively improbable.

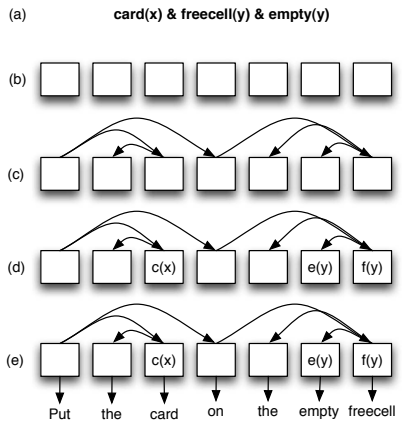


Figure 2: A graphical depiction of the generative process by which sentences are produced from formulae

3 Generative Model

These intuitions are formalized in a generative account of how sentences are stochastically produced from a set of logical formulae. This generative story guides an inference procedure for recovering logical formulae that are likely to have generated any observed set of texts, which is described in Section 4.

The outline of the generative process is depicted in Figure 2. For each sentence, we begin in step (a) by drawing a formula f from a Dirichlet process (Ferguson, 1973). The Dirichlet process de-

fines a non-parametric mixture model, and has the effect of adaptively selecting the appropriate number of formulae to explain the observed sentences in the corpus.[1] We then draw the sentence length from some distribution over positive integers; as the sentence length is always observed, we need not define the distribution (step (b)). In step (c), a dependency tree is drawn from a uniform distribution over spanning trees with a number of nodes equal to the length of the sentence. In step (d) we draw an alignment of the literals in f to nodes in the dependency tree, written $\mathbf{a}_t(f)$. The distribution over alignments is described in Section 3.1. Finally, the aligned literals are used to generate the words at each slot in the dependency tree. A more formal definition of this process is as follows:

- Draw λ, the expected number of literals per formula, from a Gamma distribution $\mathcal{G}(u, v)$.
- Draw an infinite set of formulae $\mathbf{f}$. For each formula f_i,
  - Draw the formula length $\#|f_i|$ from a Poisson distribution, $n_i \sim \text{Poisson}(\lambda)$.
  - Draw n_i literals from a uniform distribution.
- Draw π, an infinite multinomial distribution over formulae: $\pi \sim \text{GEM}(\pi_0)$, where GEM refers to the stick-breaking prior (Sethuraman, 1994) and $\pi_0 = 1$ is the concentration parameter. By attaching the multinomial π to the infinite set of formulae $\mathbf{f}$, we create a Dirichlet process. This is conventionally written $DP(\pi_0, G_0)$, where the base distribution G_0 encodes only the distribution over the number of literals, Poisson(λ).
- For each of D documents, draw the number of sentences $T \sim$ Poisson. For each of the T sentences in the document,
  - Draw a formula $f \sim DP(\pi_0, G_0)$ from the Dirichlet Process described above.
  - Draw a sentence length $\#|s| \sim$ Poisson.
  - Draw a dependency graph t (a spanning tree of size $\#|s|$) from a uniform distribution.
  - Draw an *alignment* $\mathbf{a}_t(f)$, an injective mapping from literals in f to nodes in the dependency structure t. The distribution over alignments is described in Section 3.1.
  - Draw the sentence s from the formula f and the alignment $\mathbf{a}(f)$. For each word token $w_i \in s$ is drawn from $p(w_i|a_t(f, i))$, where $a_t(f, i)$ indicates the (possibly empty) literal assigned to slot i in the alignment $\mathbf{a}_t(f)$ (Section 3.2).

[1] There are many recent applications of Dirichlet processes in natural language processing, e.g. Goldwater et al. (2006).

3.1 Distribution over Alignments

The distribution over alignments reflects our intuition that when literals share variables, they will be aligned to word slots that are nearby in the dependency structure; literals that do not share variables should be more distant. This is formalized by applying the concepts of excess terms and shared terms defined in Section 2. After computing the number of excess and shared terms in each subtree t_i, we can compute a local score (LS) for that subtree:

$$LS(\mathbf{a}_t(f); t_i) = \alpha \cdot NShared(\mathbf{a}_t(f), t_i) + \beta \cdot NExcess(\mathbf{a}_t(f), t_i) \cdot height(t_i).$$

This scoring function can be applied recursively to each subtree in t; the overall score of the tree is the recursive sum,

$$score(\mathbf{a}_t(f); t) = LS(\mathbf{a}_t(f); t) + \sum_i^n score(\mathbf{a}_t(f); t_i), \quad (1)$$

where t_i indicates the i^{th} subtree of t. We hypothesize a generative process that produces all possible alignments, scores them using $score(\mathbf{a}_t(f); t)$, and selects an alignment with probability,

$$p(\mathbf{a}_t(f)) \propto \exp\{-score(\mathbf{a}_t(f); t)\}. \quad (2)$$

In our experiments, we define the parameters $\alpha = 1, \beta = 1$.

3.2 Generation of Lexical Items

Once the logical formula is aligned to the parse structure, the generation of the lexical items in the sentence is straightforward. For word slots to which no literals are aligned, the lexical item is drawn from a language model θ, estimated from the entire document collection. For slots to which at least one literal is aligned, we construct a language model ϕ in which the probability mass is divided equally among all glosses of aligned predicates. The language model θ is used as a backoff, so that there is a strong bias in favor of generating glosses, but some probability mass is reserved for the other lexical items.

4 Inference

This section describes a sampling-based inference procedure for obtaining a set of formulae $\mathbf{f}$ that explain the observed text $\mathbf{s}$ and dependency structures $\mathbf{t}$. We perform Gibbs sampling over the formulae assigned to each sentence. Using the Chinese Restaurant Process interpretation of the Dirichlet Process (Aldous, 1985), we marginalize π, the infinite multinomial over all possible formulae: at each sampling step we select either an existing formula, or stochastically generate a new formula. After each full round of Gibbs sampling, a set of Metropolis-Hastings moves are applied to explore modifications of the formulae. This procedure converges on a stationary Markov chain centered on a set of formulae that cohere well with the lexical and syntactic properties of the text.

4.1 Assigning Sentences to Formulae

For each sentence s_i and dependency tree t_i, a hidden variable y_i indicates the index of the formula that generates the text. We can resample y_i using Gibbs sampling. In the non-parametric setting, y_i ranges over all non-negative integers; the Chinese Restaurant Process formulation marginalizes the infinite-dimensional parameter π, yielding a prior based on the counts for each "active" formula (to which at least one other sentence is assigned), and a pseudo-count representing all non-active formulae. Given K formulae, the prior on selecting formula j is:

$$p(y_i = j|\mathbf{y}_{-i}, \pi_0) \propto \begin{cases} \mathbf{n}_{-i}(j) & j < K \\ \pi_0 & j = K, \end{cases} \quad (3)$$

where $\mathbf{y}_{-i}$ refers to the assignments of all y other than y_i and $\mathbf{n}_{-i}$ refers to the counts over these assignments. Each $j < K$ identifies an existing formula in $\mathbf{f}$, to which at least one other sentence is assigned. When $j = K$, this means a new formula f^* must be generated.

To perform Gibbs sampling, we draw from the posterior distribution over y_i,

$$p(y_i|s_i, t_i\mathbf{f}, f^*, \mathbf{y}_{-i}, \pi_0) \propto p(y_i|\mathbf{y}_{-i}, \pi_0)p(s_i, t_i|\mathbf{y}_i, \mathbf{f}, f^*),$$

where the first term is the prior defined in Equation 3 and the latter term is the likelihood of generating the parsed sentence $\langle s_i, t_i \rangle$ from the formula indexed by y_i.

To compute the probability of a parsed sentence given a formula, we sum over alignments,

$$\begin{aligned} p(s, t|f) &= \sum_{\mathbf{a}_t(f)} p(s, t, \mathbf{a}_t(f)|f) \\ &= \sum_{\mathbf{a}_t(f)} p(s|\mathbf{a}_t(f))p(t, \mathbf{a}_t(f)|f) \\ &= \sum_{\mathbf{a}_t(f)} p(s|\mathbf{a}_t(f))p(\mathbf{a}_t(f)|t, f)p(t|f), \end{aligned} \quad (4)$$

applying the chain rule and independence assumptions from the generative model. The result is a product of three terms: the likelihood of the lexical items given the aligned predicates (defined in Section 3.2; the likelihood of the alignment given the dependency tree and formula (defined in equation 2), and the probability of the dependency tree given the formula, which is uniform.

Equation 4 takes a sum across alignments, but most of the probability mass of $p(s|\mathbf{a}_t(f))$ will be concentrated on alignments in which predicates cover words that gloss them. Thus, we can apply an approximation,

$$p(s, t|f) \approx \sum_{\mathbf{a}_t(f)}^{N} p(s|\mathbf{a}_t(f))p(\mathbf{a}_t(f)|t, f)p(t|f), \quad (5)$$

in which we draw N samples in which predicates are aligned to their glosses whenever possible.

Similarly, Equation 2 quantifies the likelihood of an alignment only to a constant of proportionality; again, a sum over possible alignments is necessary. We do not expect the prior on alignments to be strongly peaked like the sentence likelihood, so we approximate the normalization term by sampling M alignments at random and extrapolating:

$$\begin{aligned} p(\mathbf{a}_t(f)|t, f) &\propto q(\mathbf{a}_t(f); t) \\ &= \frac{q(\mathbf{a}_t(f); t)}{\sum_{\mathbf{a}'_t(f)} q(\mathbf{a}'_t(f); t)} \\ &\approx \frac{\#|\mathbf{a}'_t(f)|}{M} \frac{q(\mathbf{a}_t(f); t)}{\sum_{\mathbf{a}'_t(f)}^{M} q(\mathbf{a}'_t(f); t)}, \end{aligned}$$

where $q(\mathbf{a}_t(f); t) = \exp\{-score(\mathbf{a}_t(f); t)\}$, defined in Equation 2. In our experiments, we set N to at most 10, and $M = 20$. Drawing larger numbers of samples had no discernible effect on system output.

4.1.1 Generating new formulae

Chinese Restaurant Process sampling requires the generation of new candidate formulae at each resampling stage. To generate a new formula, we first sample the number of literals. As described in the generative story (Section 3), the number of literals is drawn from a Poisson distribution with parameter θ. We treat θ as unknown and marginalize, using the Gamma hyperprior $\mathcal{G}(u, v)$. Due to Poisson-Gamma conjugacy, this marginalization can be performed analytically, yielding a Negative-Binomial distribution with parameters $\langle u + \sum_i \#|f_i|, (1 + K + v)^{-1} \rangle$, where $\sum_i \#|f_i|$ is the sum of the number of literals in each formula, and K is the number of formulae which generate at least one sentence. In this sense, the hyperpriors u and v act as pseudo counts. We set $u = 3, v = 1$, reflecting a weak prior expectation of three literals per predicate.

After drawing the size of the formula, the predicates are selected from a uniform random distribution. Finally, the terms are assigned: at each slot, we reuse a previous term with probability 0.5, unless none is available; otherwise a new term is generated.

4.2 Proposing changes to formulae

The assignment resampling procedure has the ability to generate new formulae, thus exploring the space of relational features. However, to explore this space more rapidly, we introduce four Metropolis-Hastings moves that modify existing formulae (Gilks, 1995): adding a literal, deleting a literal, substituting a literal, and rearranging the terms of the formula. For each proposed move, we recompute the joint likelihood of the formula and all aligned sentences. The move is stochastically accepted based on the ratio of the joint likelihoods of the new and old configurations, multiplied by a Hastings correction.

The joint likelihood with respect to formula f is computed as $p(\mathbf{s}, \mathbf{t}, f) = p(f) \prod_i p(s_i, t_i|f)$. The prior on f considers only the number of literals, using a Negative-Binomial distribution as described in section 4.1.1. The likelihood $p(s_i, t_i|f)$ is given in equation 4. The Hastings correction is $\tilde{p}(f' \rightarrow f)/\tilde{p}(f \rightarrow f')$, with $\tilde{p}(f \rightarrow f')$ indicating the probability of proposing a move from f to f',and $\tilde{p}(f' \rightarrow f)$ indicating the probability of proposing the reverse move. The Hastings corrections depend on the arity of the predicates being added and removed; the derivation is straightforward but tedious. We plan to release a technical report with complete details.

4.3 Summary of inference

The final inference procedure iterates between Gibbs sampling of assignments of formulae to sentences, and manipulating the formulae through Metropolis-Hastings moves. A full iteration comprises proposing a move to each formula, and then using Gibbs sampling to reconsider all assignments. If a formula no longer has any sentences assigned to it, then it is dropped from the active set, and can no longer be selected in Gibbs sampling – this is standard in the Chinese Restaurant Process.

Five separate Markov chains are maintained in parallel. To allow the sampling procedure to converge to a stationary distribution, each chain begins with 100 iterations of "burn-in" sampling, without storing the output. At this point, we perform another 100 iterations, storing the state at the end of each iteration.[2] All formulae are ranked according to the cumulative number of sentences to which they are assigned (across all five Markov chains), aggregating the counts for multiple instances of identical formulae. This yields a ranked list of formulae which will be used in our framework as features for relational learning.

5 Evaluation

Our experimental setup is designed to evaluate the quality of the semantic abstraction performed by our model. The logical formulae obtained by our system are applied as features for relational learning of the rules of the game of Freecell solitaire. We investigate whether these features enable better generalization given varying number of training examples of Freecell game states. We also quantify the specific role of syntax, lexical choice, and feature expressivity in learning performance. This section describes the details of this evaluation.

5.1 Relational Learning

We perform relational learning using Inductive Logic Programming (ILP), which constructs generalized rules by assembling smaller logical formulae to explain observed propositional examples (Muggleton, 1995). The lowest level formulae consist of basic sensors that describe the environment. ILP's expressivity enables it to build complex conjunctions of these building blocks, but at the cost of tractability. Our evaluation asks whether the logical formulae abstracted from text

[2] Sampling for more iterations was not found to affect performance on development data, and the model likelihood appeared stationary after 100 iterations.

Predicate	Glosses
card(x)	card
tableau(x)	column, tableau
freecell(x)	freecell, cell
homecell(x)	foundation, cell, homecell
value(x,y)	ace, king, rank, 8, 3, 7, lowest, highest
successor(x,y)	higher, sequence, sequential
color(x,y)	black, red, color
suit(x,y)	suit, club, diamond, spade, heart
on(x,y)	onto
top(x,y)	bottom, available, top
empty(x)	empty

Table 1: Predicates in the Freecell world model, with natural language glosses obtained from the development set text.

can transform the representation to facilitate learning. We compare against both the sensor-level representation as well as richer representations that do not benefit from the full power of our model's semantic analysis.

The ALEPH[3] ILP system, which is primarily based on PROGOL (Muggleton, 1995), was used to induce the rules of game. The search parameters remained constant for all experiments.

5.2 Resources

There are four types of resources required to work in the reading-to-learn setting: a world model, instructional text, a small set of glosses that map from text to elements of the world model, and labeled examples of correct and incorrect actions in the world. In our experiments, we consider the domain of Freecell solitaire, a popular card game (Morehead and Mott-Smith, 1983) in which cards are moved between various types of locations, depending on their suit and rank. We now describe the resources for the Freecell domain in more detail.

World Model Freecell solitaire can be described formally using first order logic; we consider a slightly modified version of the representation from the Planning Domain Definition Language (PDDL), which is used in automatic game-playing competitions. Specifically, there are 87 constants: 52 cards, 16 locations, 13 values, four suits, and two colors. These constants are combined with a fixed set of 11 predicates, listed in Table 1.

Instructional Text Our approach relies on text that describes how to operate in the Freecell solitaire domain. A total of five instruction sets were obtained from the Internet. Due to the popularity of the Microsoft implementation of Freecell, instructions often contain information specific to playing Freecell on a computer. We manually removed sentences which did not focus on the card aspects of Freecell (e.g., how to set up the board and information regarding where to click to move cards). In order to use our semantic abstraction model, the instructions were part-of-speech tagged with the Stanford POS Tagger (Toutanova and Manning, 2000) and dependency parses were obtained using Malt (Nivre, 2006).

Glosses Our reading to learn setting requires a small set of glosses, which are surface forms commonly used to represent predicates from the world model. We envision an application scenario in which a designer manually specifies a few glosses for each predicate. However, for the purposes of evaluation, it would be unprincipled for the experimenters to handcraft the ideal set of glosses. Instead, we gathered a development set of text and annotated the lexical mentions of the world model predicates in text. This annotation is used to obtain glosses to apply to the evaluation text. This approximates a scenario in which the designer has a reasonable idea of how the domain will be described in text, but no prior knowledge of the specific details of the text instructions. Our experiments used glosses that occurred two or more times in the instructions: this yields a total of 32 glosses for 11 predicates, as shown in Table 1.

Evaluation game data Ultimately, the semantic abstraction obtained from the text is applied to learning on labeled examples of correct and incorrect actions in the world model. For evaluation, we automatically generated a set of *move scenarios*: game states with one positive example (a legal move) and one negative example (an illegal move). To avoid bias in the data we generate an equal number of move scenarios from each of three types: moves to the freecells, homecells, and tableaux. For our experiments we vary the number of move scenarios in the training set; the development and test sets consist of 900 and 1500 move scenarios respectively.

5.3 Evaluation Settings

We compare four different feature sets, which will be provided to the ALEPH ILP learner. All feature sets include the sensor-level predicates shown in Table 1. The FULL-MODEL feature set also includes the top logical formulae obtained in our model's semantic abstract (see Sec-

[3]Freely available from http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/

tion 4.3). The NO-SYNTAX feature set is obtained from a variant of our model in which the influence of syntax is removed by setting parameters $\alpha, \beta = 0$. The SENSORS-ONLY feature set uses only the sensor-level predicates. Finally, the RELATIONAL-RANDOM feature set is constructed by replacing each feature in the FULL-MODEL set with a randomly generated relational feature of identical expressivity (each predicate is replaced by a randomly chosen alternative with identical arity; terms are also assigned randomly). This ensures that any performance gains obtained by our model were not due merely to the greater expressivity of its relational features. The number of features included in each scenario is tuned on a development set of test examples.

The performance metric assesses the ability of the ILP learner to classify proposed Freecell moves as legal or illegal. As the evaluation set contains an equal number of positive and negative examples, accuracy is the appropriate metric. The training scenarios are randomly generated; we repeat each run 50 times and average our results. For the RELATIONAL-RANDOM feature set – in which predicates and terms are chosen randomly – we also regenerate the formulae per run.

6 Results

Table 2 shows a comparison of the results using the setup described above. Our FULL-MODEL achieves the best performance at every training set size, consistently outperforming the SENSORS-ONLY representation by an absolute difference of three to four percent. This demonstrates the semantic abstract obtained by our model does indeed facilitate machine learning in this domain.

RELATIONAL-RANDOM provides a baseline of relational features with equal expressivity to those chosen by our model, but with the predicates and terms selected randomly. We consistently outperform this baseline, demonstrate that the improvement obtained over the sensors only representation is not due merely to the added expressivity of our features.

The third row compares against NO-SYNTAX, a crippled version of our model that incorporates lexical features but not the syntactic structure. The results are stronger than the SENSORS-ONLY and RELATIONAL-RANDOM baselines, but still weaker than our full system. This demonstrates the syntactic features incorporated by our model result in better semantic representations of the underlying text.

Features	Number of training scenarios			
	15	30	60	120
SENSORS-ONLY	79.12	88.07	92.77	93.73
RELATIONAL-RANDOM	82.72	89.14	93.08	94.17
NO-SYNTAX	80.98	89.79	94.11	97.04
FULL-MODEL	**82.89**	**91.00**	**95.23**	**97.45**

Table 2: Results as number of training examples varied. Each value represents the accuracy of the induced rules obtained with the given feature set.

$\texttt{card}(x_1) \wedge \texttt{tableau}(x_2)$
$\texttt{card}(x_1) \wedge \texttt{freecell}(x_2)$
$\texttt{homecell}(x_1) \wedge \texttt{value}(x_2,x_3)$
$\texttt{empty}(x_1) \wedge \texttt{freecell}(x_1)$
$\texttt{card}(x_1) \wedge \texttt{top}(x_1,x_2)$
$\texttt{card}(x_1) \wedge \texttt{homecell}(x_2)$
$\texttt{freecell}(x_1) \wedge \texttt{homecell}(x_2)$
$\texttt{card}(x_1) \wedge \texttt{tableau}(x_1)$
$\texttt{card}(x_1) \wedge \texttt{top}(x_2,x_1)$
$\texttt{homecell}(x_1)$
$\texttt{card}(x_1) \wedge \texttt{homecell}(x_1)$
$\texttt{color}(x_1,x_2) \wedge \texttt{value}(x_3,x_4)$
$\texttt{suit}(x_1,x_2) \wedge \texttt{value}(x_3,x_4)$
$\texttt{value}(x_1,x_2) \wedge \texttt{value}(x_3,x_4)$
$\texttt{homecell}(x_1) \wedge \texttt{successor}(x_2,x_3)$

Figure 3: The top 15 features recovered by the semantic abstraction of our full model.

Figure 3 shows the top 15 formulae recovered by the full model running on the evaluation text. Features such as $\texttt{empty}(x_1) \wedge \texttt{freecell}(x_1)$ are useful because they reuse variables to ensure that objects have key properties – in this case, ensuring that a freecell is empty. Other features, such as $\texttt{homecell}(x_1) \wedge \texttt{value}(x_2, x_3)$, help to focus the search on useful conjunctions of predicates (in Freecell, the legality of playing a card on a homecell depends on the value of the card). Note that three of these 15 formulae are trivially useless, in that they are always false: e.g., $\texttt{card}(x_1) \wedge \texttt{tableau}(x_1)$. This illustrates the importance of term assignment in obtaining useful features for learning. In the NO-SYNTAX system, which ignores the relationship between term assignment and syntactic structure, eight of the top 15 formulae were trivially useless due to term incompatibility.

7 Related Work

This paper draws on recent literature on extracting logical forms from surface text (Zettlemoyer and Collins, 2005; Ge and Mooney, 2005; Downey et al., 2005; Liang et al., 2009), interpreting language in the context of a domain (Chen and Mooney, 2008), and using an actionable domain to guide text interpretation (Branavan et al., 2009). We differentiate our research in several dimensions:

Language Interpretation Instructional text describes generalized statements about entities in the domain and the way they interact, thus the text does not correspond directly to concrete sensory inputs in the world (i.e., a specific world state). Our interpretation captures these generalizations as first-order logic statements that can be evaluated given a specific state. This contrasts to previous work which interprets "directions" and thus assumes a direct correspondence between text and world state (Branavan et al., 2009; Chen and Mooney, 2008).

Supervision Our work avoids supervision in the form of labeled examples, using only a minimal set of natural language glosses per predicate. Previous work also considered the supervision signal obtained by interpreting natural language in the context of a formal domain. Branavan et al. (2009) use feedback from a world model as a supervision signal. Chen and Mooney (2008) use temporal alignment of text and grounded descriptions of the world state. In these approaches, concrete domain entities are grounded in language interpretation, and therefore require only a propositional semantic representation. Previous approaches for interpreting generalized natural language statements are trained from labeled examples (Zettlemoyer and Collins, 2005; Lu et al., 2008).

Level of analysis We aim for an *abstractive semantic summary* across multiple documents, whereas other approaches attempt to produce logical forms for individual sentences (Zettlemoyer and Collins, 2005; Ge and Mooney, 2005). We avoid the requirement that each sentence have a meaningful interpretation within the domain, allowing us to handle relatively unstructured text.

Evaluation We do not evaluate the representations obtained by our model; rather we assess whether these representations improve learning performance. This is similar to work on GeoQuery (Wong and Mooney, 2007; Ge and Mooney, 2005), and also to recent work on following step-by-step directions (Branavan et al., 2009). While these evaluations are performed on the basis of individual sentences, actions, or system responses, we evaluate the holistic semantic analysis obtained by our system.

Model We treat surface text as generated from a latent semantic description. Lu et al. (2008) apply a generative model, but require a complete derivation from semantics to the lexical representation, while we favor a more flexible semantic analysis that can be learned without annotation and applied to noisy text. More similar is the work of Liang et al. (2009), which models the generation of semantically-relevant fields using lexical and discourse features. Our approach differs by accounting for syntax, which enables a more expressive semantic representation that includes ungrounded variables.

Relational learning The output of our semantic analysis is applied to learning in a structured relational space, using ILP. A key difficulty with ILP is that the increased expressivity dramatically expands the hypothesis space, and it is widely agreed that some learning bias is required for ILP to be tractable (Nédellec et al., 1996; Cumby and Roth, 2003). Our work can be viewed as a new method for acquiring such bias from text; moreover, our approach is not specialized for ILP and may be used to transform the feature space in other forms of relational learning as well (Roth and Yih, 2001; Cumby and Roth, 2003; Richardson and Domingos, 2006).

8 Conclusion

This paper demonstrates a new setting for semantic analysis, which we term *reading to learn*. We handle text which describes the world in general terms rather than refereing to concrete entities in the domain. We obtain a semantic abstract of multiple documents, using a novel, minimally-supervised generative model that accounts for both syntax and lexical choice. The semantic abstract is represented as a set of predicate logic formulae, which are applied as higher-order features for learning. We demonstrate that these features improve learning performance, and that both the lexical and syntactic aspects of our model yield substantial contributions.

In the current setup, we produce an "overgenerated" semantic representation comprised of useful features for learning but also some false positives. Learning in our system can be seen as the process of pruning this representation by selecting useful formulae based on interaction with the training data. In the future we hope to explore ways to interleave semantic analysis with exploration of the learning domain, by using the environment as a supervision signal for linguistic analysis.

Acknowledgments We thank Gerald DeJong, Julia Hockenmaier, Alex Klementiev and the anonymous reviewers for their helpful feedback. This work is supported by DARPA funding under the Bootstrap Learning Program and the Beckman Institute Postdoctoral Fellowship.

References

Aldous, David J. 1985. Exchangeability and related topics. *Lecture Notes in Math* 1117:1–198.

Branavan, S. R. K., Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing Processing (ACL-IJCNLP 2009)*. Singapore.

Chen, David L. and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of 25th International Conference on Machine Learning (ICML 2008)*. Helsinki, Finland, pages 128–135.

Cumby, Chad and Dan Roth. 2003. On kernel methods for relational learning. In *Proceedings of the Twentieth International Conference (ICML 2003)*. Washington, DC, pages 107–114.

Downey, Doug, Oren Etzioni, and Stephen Soderland. 2005. A probabilistic model of redundancy in information extraction. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2005)*. pages 1034–1041.

Ferguson, Thomas S. 1973. A bayesian analysis of some nonparametric problems. *The Annals of Statistics* 1(2):209–230.

Ge, Ruifang and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*. Ann Arbor, MI, pages 128–135.

Gilks, Walter R. 1995. *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC.

Goldwater, Sharon, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*. Sydney, Australia, pages 673–680.

Liang, Percy, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing Processing (ACL-IJCNLP 2009)*. Singapore.

Lu, Wei, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP 2008)*. Honolulu, Hawaii, pages 783–792.

Morehead, Albert H. and Geoffrey Mott-Smith. 1983. *The Complete Book of Solitaire and Patience Games*. Bantam.

Muggleton, Stephen. 1995. Inverse entailment and progol. *New Generation Computing Journal* 13:245–286.

Nédellec, C., C. Rouveirol, H. Adé, F. Bergadano, and B. Tausend. 1996. Declarative bias in ILP. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, IOS Press, pages 82–103.

Nivre, Joakim. 2006. *Inductive dependency parsing*. Springer.

Richardson, Matthew and Pedro Domingos. 2006. Markov logic networks. *Machine Learning* 62:107–136.

Roth, Dan and Wen-tau Yih. 2001. Relational learning via propositional algorithms: An information extraction case study. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2001)*. pages 1257–1263.

Sethuraman, Jayaram. 1994. A constructive definition of dirichlet priors. *Statistica Sinica* 4(2):639–650.

Toutanova, Kristina and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*. pages 63–70.

Wong, Yuk Wah and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*. Prague, Czech Republic, pages 128–135.

Zettlemoyer, Luke S. and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005)*. pages 658–666.

Supervised Models for Coreference Resolution

Altaf Rahman and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{altaf,vince}@hlt.utdallas.edu

Abstract

Traditional learning-based coreference resolvers operate by training a *mention-pair* classifier for determining whether two mentions are coreferent or not. Two independent lines of recent research have attempted to improve these mention-pair classifiers, one by learning a *mention-ranking* model to rank preceding mentions for a given anaphor, and the other by training an *entity-mention* classifier to determine whether a preceding cluster is coreferent with a given mention. We propose a cluster-ranking approach to coreference resolution that combines the strengths of mention rankers and entity-mention models. We additionally show how our cluster-ranking framework naturally allows discourse-new entity detection to be learned jointly with coreference resolution. Experimental results on the ACE data sets demonstrate its superior performance to competing approaches.

1 Introduction

Noun phrase (NP) coreference resolution is the task of identifying which NPs (or *mentions*) refer to the same real-world entity or concept. Traditional learning-based coreference resolvers operate by training a model for classifying whether two mentions are co-referring or not (e.g., Soon et al. (2001), Ng and Cardie (2002b), Kehler et al. (2004), Ponzetto and Strube (2006)). Despite their initial successes, these *mention-pair* models have at least two major weaknesses. First, since each candidate antecedent for a mention to be resolved (henceforth an *active mention*) is considered independently of the others, these models only determine how good a candidate antecedent is relative to the active mention, but not how good a candidate antecedent is relative to other candidates. In other words, they fail to answer the critical question of which candidate antecedent is most probable. Second, they have limitations in their expressiveness: the information extracted from the two mentions alone may not be sufficient for making an informed coreference decision, especially if the candidate antecedent is a pronoun (which is semantically empty) or a mention that lacks descriptive information such as gender (e.g., *Clinton*).

To address the first weakness, researchers have attempted to train a *mention-ranking* model for determining which candidate antecedent is most probable given an active mention (e.g., Denis and Baldridge (2008)). Ranking is arguably a more natural reformulation of coreference resolution than classification, as a ranker allows all candidate antecedents to be considered *simultaneously* and therefore directly captures the competition among them. Another desirable consequence is that there exists a natural resolution strategy for a ranking approach: a mention is resolved to the candidate antecedent that has the highest rank. This contrasts with classification-based approaches, where many clustering algorithms have been employed to co-ordinate the pairwise coreference decisions (because it is unclear which one is the best).

To address the second weakness, researchers have investigated the acquisition of *entity-mention* coreference models (e.g., Luo et al. (2004), Yang et al. (2004)). Unlike mention-pair models, these entity-mention models are trained to determine whether an active mention belongs to a preceding, possibly partially-formed, coreference cluster. Hence, they can employ *cluster-level* features (i.e., features that are defined over any subset of mentions in a preceding cluster), which makes them more expressive than mention-pair models.

Motivated in part by these recently developed models, we propose in this paper a *cluster-ranking* approach to coreference resolution that combines the strengths of mention-ranking mod-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 968–977,
Singapore, 6-7 August 2009.

els and entity-mention models. Specifically, we recast coreference as the problem of determining which of a set of preceding coreference *clusters* is the best to link to an active mention using a learned *cluster ranker*. In addition, we show how discourse-new detection (i.e., the task of determining whether a mention introduces a new entity in a discourse) can be learned *jointly* with coreference resolution in our cluster-ranking framework. It is worth noting that researchers typically adopt a *pipeline* coreference architecture, performing discourse-new detection prior to coreference resolution and using the resulting information to prevent a coreference system from resolving mentions that are determined to be discourse-new (see Poesio et al. (2004) for an overview). As a result, errors in discourse-new detection could be propagated to the resolver, possibly leading to a deterioration of coreference performance (see Ng and Cardie (2002a)). Jointly learning discourse-new detection and coreference resolution can potentially address this error-propagation problem.

In sum, we believe our work makes three main contributions to coreference resolution:

Proposing a simple, yet effective coreference model. Our work advances the state-of-the-art in coreference resolution by bringing learning-based coreference systems to the next level of performance. When evaluated on the ACE 2005 coreference data sets, cluster rankers outperform three competing models — mention-pair, entity-mention, and mention-ranking models — by a large margin. Also, our joint-learning approach to discourse-new detection and coreference resolution consistently yields cluster rankers that outperform those adopting the pipeline architecture. Equally importantly, cluster rankers are conceptually simple and easy to implement and do not rely on sophisticated training and inference procedures to make coreference decisions in dependent relation to each other, unlike relational coreference models (see McCallum and Wellner (2004)).

Bridging the gap between machine-learning approaches and linguistically-motivated approaches to coreference resolution. While machine learning approaches to coreference resolution have received a lot of attention since the mid-90s, popular learning-based coreference frameworks such as the mention-pair model are arguably rather unsatisfactory from a linguistic point of view. In particular, they have not leveraged advances in discourse-based anaphora resolution research in the 70s and 80s. Our work bridges this gap by realizing in a new machine learning framework ideas rooted in Lappin and Leass's (1994) heuristic-based pronoun resolver, which in turn was motivated by classic salience-based approaches to anaphora resolution.

Revealing the importance of adopting the right model. While entity-mention models have previously been shown to be worse or at best marginally better than their mention-pair counterparts (Luo et al., 2004; Yang et al., 2008), our cluster-ranking models, which are a natural extension of entity-mention models, significantly outperformed all competing approaches. This suggests that the use of an appropriate learning framework can bring us a long way towards high-performance coreference resolution.

The rest of the paper is structured as follows. Section 2 discusses related work. Section 3 describes our baseline coreference models: mention-pair, entity-mention, and mention-ranking. We discuss our cluster-ranking approach in Section 4, evaluate it in Section 5, and conclude in Section 6.

2 Related Work

Heuristic-based cluster ranking. As mentioned previously, the work most related to ours is Lappin and Leass (1994), whose goal is to perform pronoun resolution by assigning an anaphoric pronoun to the highest-scored preceding cluster. Nevertheless, Lappin and Leass's work differs from ours in several respects. First, they only tackle pronoun resolution rather than the full coreference task. Second, their algorithm is heuristic-based; in particular, the score assigned to a preceding cluster is computed by summing over the weights associated with the factors applicable to the cluster, where the weights are determined heuristically, rather than learned, unlike ours.

Like many heuristic-based pronoun resolvers (e.g., Mitkov (1998)), they first apply a set of constraints to filter grammatically incompatible candidate antecedents and then rank the remaining ones using salience factors. As a result, their cluster-ranking model employs only factors that capture the salience of a cluster, and can therefore be viewed as a simple model of attentional state (see Grosz and Sidner (1986)) realized by coreference clusters. By contrast, our resolution strategy is learned without applying hand-coded con-

straints in a separate filtering step. In particular, we attempt to determine the compatibility between a cluster and an active mention, using factors that determine not only salience (e.g., the distance between the cluster and the mention) but also lexical and grammatical compatibility, for instance.

Entity-mention coreference models. Luo et al. (2004) represent one of the earliest attempts to investigate learning-based entity-mention models. They use the ANY predicate to generate cluster-level features as follows: given a binary-valued feature X defined over a pair of mentions, they introduce an ANY-X cluster-level feature, which has the value TRUE if X is true between the active mention and *any* mention in the preceding cluster under consideration. Contrary to common wisdom, this entity-mention model underperforms its mention-pair counterpart in spite of the generalization from mention-pair to cluster-level features.

In Yang et al.'s (2004) entity-mention model, a training instance is composed of an active mention m_k, a preceding cluster C, and a mention m_j in C that is closest in distance to m_k in the associated text. The feature set used to represent the instance is primarily composed of features that describe the relationship between m_j and m_k, as well as a few cluster-level features. In other words, the model still relies heavily on features used in a mention-pair model. In particular, the inclusion of m_j in the feature vector representation to some extent reflects the authors' lack of confidence that a strong entity-mention model can be trained without mention-pair-based features. Our ranking model, on the other hand, is trained without such features. More recently, Yang et al. (2008) have proposed another entity-mention model trained by inductive logic programming. Like their previous work, the scarcity of cluster-level predicates (only two are used) under-exploits the expressiveness of entity-mention models.

Mention ranking. The notion of ranking candidate antecedents can be traced back to centering algorithms, many of which use grammatical roles to rank forward-looking centers (see Grosz et al. (1995), Walker et al. (1998), and Mitkov (2002)). However, mention ranking has been employed in learning-based coreference resolvers only recently. As mentioned before, Denis and Baldridge (2008) train a mention-ranking model. Their work can be viewed as an extension of Yang et al.'s (2003) twin-candidate coreference model, which ranks only two candidate antecedents at a time. Unlike ours, however, their model ranks mentions rather than clusters, and relies on an independently-trained discourse-new detector.

Discourse-new detection. Discourse-new detection is often tackled independently of coreference resolution. Pleonastic *it*s have been detected using heuristics (e.g., Kennedy and Boguraev (1996)) and learning-based techniques such as rule learning (e.g., Müller (2006)), kernels (e.g., Versley et al. (2008)), and distributional methods (e.g., Bergsma et al. (2008)). Non-anaphoric definite descriptions have been detected using heuristics (e.g., Vieira and Poesio (2000)) and unsupervised methods (e.g., Bean and Riloff (1999)). General discourse-new detectors that are applicable to different types of NPs have been built using heuristics (e.g., Byron and Gegg-Harrison (2004)) and modeled generatively (e.g., Elsner and Charniak (2007)) and discriminatively (e.g., Uryupina (2003)). There have also been attempts to perform joint *inference* for discourse-new detection and coreference resolution using integer linear programming (ILP), where a discourse-new classifier and a coreference classifier are trained *independently* of each other, and then ILP is applied as a post-processing step to jointly infer discourse-new and coreference decisions so that they are consistent with each other (e.g., Denis and Baldridge (2007)). Joint inference is different from our joint-learning approach, which allows the two tasks to be learned jointly and not independently.

3 Baseline Coreference Models

In this section, we describe three coreference models that will serve as our baselines: the mention-pair model, the entity-mention model, and the mention-ranking model. For illustrative purposes, we will use the text segment shown in Figure 1. Each mention m in the segment is annotated as $[m]^{cid}_{mid}$, where mid is the mention id and cid is the id of the cluster to which m belongs. As we can see, the mentions are partitioned into four sets, with *Barack Obama*, *his*, and *he* in one cluster, and each of the remaining mentions in its own cluster.

3.1 Mention-Pair Model

As noted before, a mention-pair model is a classifier that decides whether or not an active mention m_k is coreferent with a candidate antecedent m_j. Each instance $\boldsymbol{i}(m_j, m_k)$ represents m_j and

[*Barack Obama*]$_1^1$ *nominated* [*Hillary Rodham Clinton*]$_2^2$ *as* [[*his*]$_3^1$ *secretary of state*]$_4^3$ *on* [*Monday*]$_5^4$. [*He*]$_6^1$ ...

Figure 1: An illustrative example

m_k and consists of the 39 features shown in Table 1. These features have largely been employed by state-of-the-art learning-based coreference systems (e.g., Soon et al. (2001), Ng and Cardie (2002b), Bengtson and Roth (2008)), and are computed automatically. As can be seen, the features are divided into four blocks. The first two blocks consist of features that describe the properties of m_j and m_k, respectively, and the last two blocks of features describe the relationship between m_j and m_k. The classification associated with a training instance is either positive or negative, depending on whether m_j and m_k are coreferent.

If one training instance were created from each pair of mentions, the negative instances would significantly outnumber the positives, yielding a skewed class distribution that will typically have an adverse effect on model training. As a result, only a subset of mention pairs will be generated for training. Following Soon et al. (2001), we create (1) a positive instance for each discourse-old mention m_k and its closest antecedent m_j; and (2) a negative instance for m_k paired with each of the intervening mentions, $m_{j+1}, m_{j+2}, \ldots, m_{k-1}$. In our running example shown in Figure 1, three training instances will be generated for *He*: ***i***(*Monday*, *He*), ***i***(*secretary of state*, *He*), and ***i***(*his*, *He*). The first two of these instances will be labeled as negative, and the last one will be labeled as positive. To train a mention-pair classifier, we use the SVM learning algorithm from the SVM^{light} package (Joachims, 2002), converting all multi-valued features into an equivalent set of binary-valued features.

After training, the resulting SVM classifier is used to identify an antecedent for a mention in a test text. Specifically, an active mention m_k selects as its antecedent the closest preceding mention that is classified as coreferent with m_k. If m_k is not classified as coreferent with any preceding mention, it will be considered discourse-new (i.e., no antecedent will be selected for m_k).

3.2 Entity-Mention Model

Unlike a mention-pair model, an entity-mention model is a classifier that decides whether or not an active mention m_k is coreferent with a *partial cluster* c_j that precedes m_k. Each training instance, ***i***(c_j, m_k), represents c_j and m_k. The features for an instance can be divided into two types: (1) features that describe m_k (i.e, those shown in the second block of Table 1), and (2) cluster-level features, which describe the relationship between c_j and m_k. Motivated by previous work (Luo et al., 2004; Culotta et al., 2007; Yang et al., 2008), we create cluster-level features from mention-pair features using four predicates: NONE, MOST-FALSE, MOST-TRUE, and ALL. Specifically, for each feature X shown in the last two blocks in Table 1, we first convert X into an equivalent set of binary-valued features if it is multi-valued. Then, for each resulting binary-valued feature X$_b$, we create four binary-valued cluster-level features: (1) NONE-X$_b$ is true when X$_b$ is false between m_k and each mention in c_j; (2) MOST-FALSE-X$_b$ is true when X$_b$ is true between m_k and less than half (but at least one) of the mentions in c_j; (3) MOST-TRUE-X$_b$ is true when X$_b$ is true between m_k and at least half (but not all) of the mentions in c_j; and (4) ALL-X$_b$ is true when X$_b$ is true between m_k and each mention in c_j. Hence, for each X$_b$, exactly one of these four cluster-level features evaluates to true.

Following Yang et al. (2008), we create (1) a positive instance for each discourse-old mention m_k and the preceding cluster c_j to which it belongs; and (2) a negative instance for m_k paired with each partial cluster whose last mention appears between m_k and its closest antecedent (i.e., the last mention of c_j). Consider again our running example. Three training instances will be generated for *He*: ***i***({*Monday*}, *He*), ***i***({*secretary of state*}, *He*), and ***i***({*Barack Obama*, *his*}, *He*). The first two of these instances will be labeled as negative, and the last one will be labeled as positive. As in the mention-pair model, we train an entity-mention classifier using the SVM learner.

After training, the resulting classifier is used to identify a preceding cluster for a mention in a test text. Specifically, the mentions are processed in a left-to-right manner. For each active mention m_k, a test instance is created between m_k and each of the preceding clusters formed so far. All the test instances are then presented to the classifier. Finally, m_k will be linked to the closest preceding cluster that is classified as coreferent with m_k. If m_k is not classified as coreferent with any

Features describing m_j, a candidate antecedent		
1	PRONOUN_1	Y if m_j is a pronoun; else N
2	SUBJECT_1	Y if m_j is a subject; else N
3	NESTED_1	Y if m_j is a nested NP; else N
Features describing m_k, the mention to be resolved		
4	NUMBER_2	SINGULAR or PLURAL, determined using a lexicon
5	GENDER_2	MALE, FEMALE, NEUTER, or UNKNOWN, determined using a list of common first names
6	PRONOUN_2	Y if m_k is a pronoun; else N
7	NESTED_2	Y if m_k is a nested NP; else N
8	SEMCLASS_2	the semantic class of m_k; can be one of PERSON, LOCATION, ORGANIZATION, DATE, TIME, MONEY, PERCENT, OBJECT, OTHERS, determined using WordNet and an NE recognizer
9	ANIMACY_2	Y if m_k is determined as HUMAN or ANIMAL by WordNet and an NE recognizer; else N
10	PRO_TYPE_2	the nominative case of m_k if it is a pronoun; else NA. E.g., the feature value for *him* is HE
Features describing the relationship between m_j, a candidate antecedent and m_k, the mention to be resolved		
11	HEAD_MATCH	C if the mentions have the same head noun; else I
12	STR_MATCH	C if the mentions are the same string; else I
13	SUBSTR_MATCH	C if one mention is a substring of the other; else I
14	PRO_STR_MATCH	C if both mentions are pronominal and are the same string; else I
15	PN_STR_MATCH	C if both mentions are proper names and are the same string; else I
16	NONPRO_STR_MATCH	C if the two mentions are both non-pronominal and are the same string; else I
17	MODIFIER_MATCH	C if the mentions have the same modifiers; NA if one of both of them don't have a modifier; else I
18	PRO_TYPE_MATCH	C if both mentions are pronominal and are either the same pronoun or different only w.r.t. case; NA if at least one of them is not pronominal; else I
19	NUMBER	C if the mentions agree in number; I if they disagree; NA if the number for one or both mentions cannot be determined
20	GENDER	C if the mentions agree in gender; I if they disagree; NA if the gender for one or both mentions cannot be determined
21	AGREEMENT	C if the mentions agree in both gender and number; I if they disagree in both number and gender; else NA
22	ANIMACY	C if the mentions match in animacy; I if they don't; NA if the animacy for one or both mentions cannot be determined
23	BOTH_PRONOUNS	C if both mentions are pronouns; I if neither are pronouns; else NA
24	BOTH_PROPER_NOUNS	C if both mentions are proper nouns; I if neither are proper nouns; else NA
25	MAXIMALNP	C if the two mentions does not have the same maximial NP projection; else I
26	SPAN	C if neither mention spans the other; else I
27	INDEFINITE	C if m_k is an indefinite NP and is not in an appositive relationship; else I
28	APPOSITIVE	C if the mentions are in an appositive relationship; else I
29	COPULAR	C if the mentions are in a copular construction; else I
30	SEMCLASS	C if the mentions have the same semantic class; I if they don't; NA if the semantic class information for one or both mentions cannot be determined
31	ALIAS	C if one mention is an abbreviation or an acronym of the other; else I
32	DISTANCE	binned values for sentence distance between the mentions
Additional features describing the relationship between m_j, a candidate antecedent and m_k, the mention to be resolved		
33	NUMBER'	the concatenation of the NUMBER_2 feature values of m_j and m_k. E.g., if m_j is *Clinton* and m_k is *they*, the feature value is SINGULAR-PLURAL, since m_j is singular and m_k is plural
34	GENDER'	the concatenation of the GENDER_2 feature values of m_j and m_k
35	PRONOUN'	the concatenation of the PRONOUN_2 feature values of m_j and m_k
36	NESTED'	the concatenation of the NESTED_2 feature values of m_j and m_k
37	SEMCLASS'	the concatenation of the SEMCLASS_2 feature values of m_j and m_k
38	ANIMACY'	the concatenation of the ANIMACY_2 feature values of m_j and m_k
39	PRO_TYPE'	the concatenation of the PRO_TYPE_2 feature values of m_j and m_k

Table 1: The feature set for coreference resolution. Non-relational features describe a mention and in most cases take on a value of **Y**ES or **N**O. Relational features describe the relationship between the two mentions and indicate whether they are **C**OMPATIBLE, **I**NCOMPATIBLE or **N**OT **A**PPLICABLE.

preceding cluster, it will be considered discourse-new. Note that all partial clusters preceding m_k are formed incrementally based on the predictions of the classifier for the first $k-1$ mentions.

3.3 Mention-Ranking Model

As noted before, a ranking model imposes a ranking on all the candidate antecedents of an active mention m_k. To train a ranker, we use the SVM ranker-learning algorithm from the SVM^{light} package. Like the mention-pair model, each training instance $\boldsymbol{i}(m_j, m_k)$ represents m_k and a preceding mention m_j. In fact, the features that represent the instance as well as the method for creating training instances are identical to those employed by the mention-pair model.

The only difference lies in the assignment of class values to training instances. Assuming that S_k is the set of training instances created for anaphoric mention m_k, the class value for an instance $\boldsymbol{i}(m_j, m_k)$ in S_k is the rank of m_j among competing candidate antecedents, which is 2 if m_j is the closest antecedent of m_k, and 1 otherwise.[1] To exemplify, consider our running example. As in the mention-pair model, three training instances will be generated for *He*: ***i***(*Monday*, *He*), ***i***(*secretary of state*, *He*), ***i***(*his*, *He*). The third instance will have a class value of 2, and the remaining two will have a class value of 1.

After training, the mention-ranking model is applied to rank the candidate antecedents for an active mention in a test text as follows. Given an active mention m_k, we follow Denis and Baldridge (2008) and use an independently-trained classifier to determine whether m_k is discourse-new. If so, m_k will not be resolved. Otherwise, we create test instances for m_k by pairing it with each of its preceding mentions. The test instances are then presented to the ranker, and the preceding mention that is assigned the largest value by the ranker is selected as the antecedent of m_k.

The discourse-new classifier used in the resolution step is trained with 26 of the 37 features[2] described in Ng and Cardie (2002a) that are deemed useful for distinguishing between anaphoric and non-anaphoric mentions. These features can be broadly divided into two types: (1) features that encode the form of the mention (e.g., NP type, number, definiteness), and (2) features that compare the mention to one of its preceding mentions.

4 Coreference as Cluster Ranking

In this section, we describe our cluster-ranking approach to NP coreference. As noted before, our approach aims to combine the strengths of entity-mention models and mention-ranking models.

4.1 Training and Applying a Cluster Ranker

For ease of exposition, we will describe in this subsection how to train and apply a cluster ranker when it is used in a pipeline architecture, where discourse-new detection is performed prior to coreference resolution. In the next subsection, we will show how the two tasks can be learned jointly.

[1] A larger class value implies a better rank in SVM^{light}.

[2] The 11 features that we did not employ are CONJ, POSSESSIVE, MODIFIER, POSTMODIFIED, SPECIAL_NOUNS, POST, SUBCLASS, TITLE, and the positional features.

Recall that a cluster ranker ranks a set of preceding clusters for an active mention m_k. Since a cluster ranker is a hybrid of a mention-ranking model and an entity-mention model, the way it is trained and applied is also a hybrid of the two. In particular, the instance representation employed by a cluster ranker is identical to that used by an entity-mention model, where each training instance ***i***(c_j, m_k) represents a preceding cluster c_j and a discourse-old mention m_k and consists of cluster-level features formed from predicates. Unlike in an entity-mention model, however, in a cluster ranker, (1) a training instance is created between each discourse-old mention m_k and *each* of its preceding clusters; and (2) since we are training a model for ranking clusters, the assignment of class values to training instances is similar to that of a mention ranker. Specifically, the class value of a training instance ***i***(c_j, m_k) created for m_k is the rank of c_j among the competing clusters, which is 2 if m_k belongs to c_j, and 1 otherwise.

Applying the learned cluster ranker to a test text is similar to applying a mention ranker. Specifically, the mentions are processed in a left-to-right manner. For each active mention m_k, we first apply an independently-trained classifier to determine if m_k is discourse-new. If so, m_k will not be resolved. Otherwise, we create test instances for m_k by pairing it with each of its preceding clusters. The test instances are then presented to the ranker, and m_k is linked to the cluster that is assigned the highest value by the ranker. Note that these partial clusters preceding m_k are formed incrementally based on the predictions of the ranker for the first $k-1$ mentions; no gold-standard coreference information is used in their formation.

4.2 Joint Discourse-New Detection and Coreference Resolution

The cluster ranker described above can be used to determine which preceding cluster a discourse-old mention should be linked to, but it cannot be used to determine whether a mention is discourse-new or not. The reason is simple: all the training instances are generated from discourse-old mentions. Hence, to jointly learn discourse-new detection and coreference resolution, we must train the ranker using instances generated from *both* discourse-old and discourse-new mentions.

Specifically, when training the ranker, we provide each active mention with the option to start

a new cluster by creating an additional instance that (1) contains features that solely describe the active mention (i.e., the features shown in the second block of Table 1), and (2) has the highest rank value among competing clusters (i.e., 2) if it is discourse-new and the lowest rank value (i.e., 1) otherwise. The main advantage of jointly learning the two tasks is that it allows the ranking model to evaluate *all* possible options for an active mention (i.e., whether to resolve it, and if so, which preceding cluster is the best) *simultaneously*.

After training, the resulting cluster ranker processes the mentions in a test text in a left-to-right manner. For each active mention m_k, we create test instances for it by pairing it with each of its preceding clusters. To allow for the possibility that m_k is discourse-new, we create an additional test instance that contains features that solely describe the active mention (similar to what we did in the training step above). All these test instances are then presented to the ranker. If the additional test instance is assigned the highest rank value by the ranker, then m_k is classified as discourse-new and will not be resolved. Otherwise, m_k is linked to the cluster that has the highest rank. As before, all partial clusters preceding m_k are formed incrementally based on the predictions of the ranker for the first $k-1$ mentions.

5 Evaluation

5.1 Experimental Setup

Corpus. We use the ACE 2005 coreference corpus as released by the LDC, which consists of the 599 training documents used in the official ACE evaluation.[3] To ensure diversity, the corpus was created by selecting documents from six different sources: Broadcast News (bn), Broadcast Conversations (bc), Newswire (nw), Webblog (wb), Usenet (un), and conversational telephone speech (cts). The number of documents belonging to each source is shown in Table 2. For evaluation, we partition the 599 documents into a training set and a test set following a 80/20 ratio, ensuring that the two sets have the same proportion of documents from the six sources.

Mention extractor. We evaluate each coreference model using both *true mentions* (i.e., gold standard mentions[4]) and *system mentions* (i.e., automatically identified mentions). To extract system mentions from a test text, we trained a mention extractor on the training texts. Following Florian et al. (2004), we recast mention extraction as a sequence labeling task, where we assign to each token in a test text a label that indicates whether it **b**egins a mention, is **i**nside a mention, or is **o**utside a mention. Hence, to learn the extractor, we create one training instance for each token in a training text and derive its class value (one of **b**, **i**, and **o**) from the annotated data. Each instance represents w_i, the token under consideration, and consists of 29 linguistic features, many of which are modeled after the systems of Bikel et al. (1999) and Florian et al. (2004), as described below.

Dataset	bn	bc	nw	wl	un	cts
# of documents	60	226	106	119	49	39

Table 2: Statistics for the ACE 2005 corpus

Lexical (7): Tokens in a window of 7: $\{w_{i-3}, \ldots, w_{i+3}\}$.

Capitalization (4): Determine whether w_i `IsAllCap`, `IsInitCap`, `IsCapPeriod`, and `IsAllLower` (see Bikel et al. (1999)).

Morphological (8): w_i's prefixes and suffixes of length one, two, three, and four.

Grammatical (1): The part-of-speech (POS) tag of w_i obtained using the Stanford log-linear POS tagger (Toutanova et al., 2003).

Semantic (1): The named entity (NE) tag of w_i obtained using the Stanford CRF-based NE recognizer (Finkel et al., 2005).

Gazetteers (8): Eight dictionaries containing pronouns (77 entries), common words and words that are not names (399.6k), person names (83.6k), person titles and honorifics (761), vehicle words (226), location names (1.8k), company names (77.6k), and nouns extracted from WordNet that are hyponyms of PERSON (6.3k).

We employ CRF++[5], a C++ implementation of conditional random fields, for training the mention detector, which achieves an F-score of 86.7 (86.1 recall, 87.2 precision) on the test set. These extracted mentions are to be used as system mentions in our coreference experiments.

Scoring programs. To score the output of a coreference model, we employ three scoring programs: MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and ϕ_3-CEAF (Luo, 2005).

[3] Since we did not participate in ACE 2005, we do not have access to the official test set.

[4] Note that only mention *boundaries* are used.

[5] Available from http://crfpp.sourceforge.net

There is a complication, however. When scoring a *response* (i.e., system-generated) partition against a *key* (i.e., gold-standard) partition, a scoring program needs to construct a mapping between the mentions in the response and those in the key. If the response is generated using true mentions, then every mention in the response is mapped to some mention in the key and vice versa; in other words, there are no *twinless* (i.e., unmapped) mentions (Stoyanov et al., 2009). However, this is not the case when system mentions are used. The aforementioned complication does not arise from the construction of the mapping, but from the fact that Bagga and Baldwin (1998) and Luo (2005) do not specify how to apply B^3 and CEAF to score partitions generated from system mentions.

We propose a simple solution to this problem: we remove all and only those twinless system mentions that are singletons before applying B^3 and CEAF. The reason is simple: since the coreference resolver has successfully identified these mentions as singletons, it should not be penalized, and removing them allows us to avoid such penalty. Note that we only remove twinless (as opposed to all) system mentions that are singletons: this allows us to reward a resolver for successful identification of singleton mentions that have twins, thus overcoming a major weakness of and common criticism against the MUC scorer. Also, we retain twinless system mentions that are non-singletons, as the resolver should be penalized for identifying spurious coreference relations. On the other hand, we do not remove twinless mentions in the key partition, as we want to ensure that the resolver makes the correct (non-)coreference decisions for them. We believe that our proposal addresses Stoyanov et al.'s (2009) problem of having very low precision when applying the CEAF scorer to score partitions of system mentions.

5.2 Results and Discussions

The mention-pair baseline. We train our first baseline, the mention-pair coreference classifier, using the SVM learning algorithm as implemented in the SVM^{light} package (Joachims, 2002).[6] Results of this baseline using true mentions and system mentions, shown in row 1 of Tables 3 and 4, are reported in terms of recall (R), precision (P), and F-score (F) provided by the three scoring programs. As we can see, this baseline achieves F-scores of 54.3–70.0 and 53.4–62.5 for true mentions and system mentions, respectively.

The entity-mention baseline. Next, we train our second baseline, the entity-mention coreference classifier, using the SVM learner. Results of this baseline are shown in row 2 of Tables 3 and 4. For true mentions, this baseline achieves an F-score of 54.8–70.7. In comparison to the mention-pair baseline, F-score rises insignificantly according to all three scorers.[7] Similar trends can be observed for system mentions, where the F-scores between the two models are statistically indistinguishable across the board. While the insignificant performance difference is somewhat surprising given the improved expressiveness of entity-mention models over mention-pair models, similar trends have been reported by Luo et al. (2004).

The mention-ranking baseline. Our third baseline is the mention-ranking coreference model, trained using the ranker-learning algorithm in SVM^{light}. To identify discourse-new mentions, we employ two methods. In the first method, we adopt a pipeline architecture, where we train an SVM classifier for discourse-new detection independently of the mention ranker on the training set using the 26 features described in Section 3.3. We then apply the resulting classifier to each test text to filter discourse-new mentions prior to coreference resolution. Results of the mention ranker are shown in row 3 of Tables 3 and 4. As we can see, the ranker achieves F-scores of 57.8–71.2 and 54.1–65.4 for true mentions and system mentions, respectively, yielding a significant improvement over the entity-mention baseline in all but one case (MUC/true mentions).

In the second method, we perform discourse-new detection jointly with coreference resolution using the method described in Section 4.2. While we discussed this joint learning method in the context of cluster ranking, it should be easy to see that the method is equally applicable to a mention ranker. Results of the mention ranker using this joint architecture are shown in row 4 of Tables 3 and 4. As we can see, the ranker achieves F-scores of 61.6–73.4 and 55.6–67.1 for true mentions and system mentions, respectively. For both types of mentions, the improvements over the corresponding results for the entity-mention baseline

[6] For this and subsequent uses of the SVM learner in our experiments, we set all parameters to their default values.

[7] We use Approximate Randomization (Noreen, 1989) for testing statistical significance, with p set to 0.05.

		MUC			CEAF			B^3		
	Coreference Model	R	P	F	R	P	F	R	P	F
1	Mention-pair model	71.7	69.2	70.4	54.3	54.3	54.3	53.3	63.6	58.0
2	Entity-mention model	71.7	69.7	70.7	54.8	54.8	54.8	53.2	65.1	58.5
3	Mention-ranking model (Pipeline)	68.7	73.9	71.2	57.8	57.8	57.8	55.8	63.9	59.6
4	Mention-ranking model (Joint)	69.4	77.8	73.4	61.6	61.6	61.6	57.0	70.1	62.9
5	Cluster-ranking model (Pipeline)	71.7	78.2	74.8	61.8	61.8	61.8	58.2	69.1	63.2
6	Cluster-ranking model (Joint)	69.9	83.3	**76.0**	63.3	63.3	**63.3**	56.0	74.6	**64.0**

Table 3: MUC, CEAF, and B^3 coreference results using true mentions.

		MUC			CEAF			B^3		
	Coreference Model	R	P	F	R	P	F	R	P	F
1	Mention-pair model	70.0	56.4	62.5	56.1	51.0	53.4	50.8	57.9	54.1
2	Entity-mention model	68.5	57.2	62.3	56.3	50.2	53.1	51.2	57.8	54.3
3	Mention-ranking model (Pipeline)	62.2	68.9	65.4	51.6	56.7	54.1	52.3	61.8	56.6
4	Mention-ranking model (Joint)	62.1	73.0	67.1	53.0	58.5	55.6	50.4	65.5	56.9
5	Cluster-ranking model (Pipeline)	65.3	72.3	68.7	54.1	59.3	56.6	55.3	63.7	59.2
6	Cluster-ranking model (Joint)	64.1	75.4	**69.3**	56.7	62.6	**59.5**	54.4	70.5	**61.4**

Table 4: MUC, CEAF, and B^3 coreference results using system mentions.

are significant, and suggest that mention ranking is a precision-enhancing device. Moreover, in comparison to the pipeline architecture in row 3, we see that F-score rises significantly by 2.2–3.8% for true mentions, and improves by a smaller margin of 0.3–1.7% for system mentions. These results demonstrate the benefits of joint modeling.

Our cluster-ranking model. Finally, we evaluate our cluster-ranking model. As in the mention-ranking baseline, we employ both the pipeline architecture and the joint architecture for discourse-new detection. Results are shown in rows 5 and 6 of Tables 3 and 4, respectively, for the two architectures. When true mentions are used, the pipeline architecture yields an F-score of 61.8–74.8, which represents a significant improvement over the mention ranker adopting the pipeline architecture. With the joint architecture, the cluster ranker achieves an F-score of 63.3–76.0. This also represents a significant improvement over the mention ranker adopting the joint architecture, the best of the baselines, and suggests that cluster ranking is a better precision-enhancing model than mention ranking. Moreover, comparing the results in these two rows reveals the superiority of the joint architecture over the pipeline architecture, particularly in terms of its ability to enhance system precision. Similar performance trends can be observed when system mentions are used.

6 Conclusions

We have presented a cluster-ranking approach that recasts the mention resolution process as the problem of finding the best preceding cluster to link an active mention to. Crucially, our approach combines the strengths of entity-mention models and mention-ranking models. Experimental results on the ACE 2005 corpus show that (1) jointly learning coreference resolution and discourse-new detection allows the cluster ranker to achieve better performance than adopting a pipeline coreference architecture; and (2) our cluster ranker significantly outperforms the mention ranker, the best of the three baseline coreference models, under both the pipeline architecture and the joint architecture. Overall, we believe that our cluster-ranking approach advances the state-of-the-art in coreference resolution both theoretically and empirically.

Acknowledgments

We thank the three anonymous reviewers for their invaluable comments on the paper. This work was supported in part by NSF Grant IIS-0812261.

References

A. Bagga and B. Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Proc. of COLING-ACL*, pages 79–85.

D. Bean and E. Riloff. 1999. Corpus-based identification of non-anaphoric noun phrases. In *Proc. of the ACL*, pages 373–380.

E. Bengtson and D. Roth. 2008. Understanding the values of features for coreference resolution. In *Proc. of EMNLP*, pages 294–303.

S. Bergsma, D. Lin, and R. Goebel. 2008. Distributional identification of non-referential pronouns. In *Proc. of ACL-08:HLT*, pages 10–18.

D. Bikel, R. Schwartz, and R. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1–3):211–231.

D. Byron and W. Gegg-Harrison. 2004. Eliminating non-referring noun phrases from coreference resolution. In *Proc. of DAARC*, pages 21–26.

A. Culotta, M. Wick, and A. McCallum. 2007. First-order probabilistic models for coreference resolution. In *Proc. of NAACL-HLT*, pages 81–88.

P. Denis and J. Baldridge. 2007. Global, joint determination of anaphoricity and coreference resolution using integer programming. In *Proc. of NAACL-HLT*, pages 236–243.

P. Denis and J. Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proc. of EMNLP*, pages 660–669.

M. Elsner and E. Charniak. 2007. A generative discourse-new model for text coherence. Technical Report CS-07-04, Brown University.

J. R. Finkel, T. Grenager, and C. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proc. of the ACL*, pages 363–370.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and I. Zitouni. 2004. A statistical model for multilingual entity detection and tracking. In *Proc. of HLT/NAACL*.

B. J. Grosz, A. K. Joshi, and S. Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–226.

B. J. Grosz and C. L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proc. of KDD*, pages 133–142.

A. Kehler, D. Appelt, L. Taylor, and A. Simma. 2004. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proc. of HLT/NAACL*.

C. Kennedy and B. Boguraev. 1996. Anaphor for everyone: Pronominal anaphora resolution without a parser. In *Proc. of COLING*, pages 113–118.

S. Lappin and H. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–562.

X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *Proc. of the ACL*, pages 135–142.

X. Luo. 2005. On coreference resolution performance metrics. In *Proc. of HLT/EMNLP*, pages 25–32.

A. McCallum and B. Wellner. 2004. Conditional models of identity uncertainty with application to noun coreference. In *Advances in NIPS*.

R. Mitkov. 2002. *Anaphora Resolution*. Longman.

R. Mitkov. 1998. Robust pronoun resolution with limited knowledge. In *Proc. of COLING/ACL*, pages 869–875.

C. Müller. 2006. Automatic detection of nonreferential it in spoken multi-party dialog. In *Proc. of EACL*, pages 49–56.

V. Ng and C. Cardie. 2002a. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proc. of COLING*, pages 730–736.

V. Ng and C. Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *Proc. of the ACL*, pages 104–111.

E. W. Noreen. 1989. *Computer Intensive Methods for Testing Hypothesis: An Introduction*. John Wiley & Sons.

M. Poesio, O. Uryupina, R. Vieira, M. Alexandrov-Kabadjov, and R. Goulart. 2004. Discourse-new detectors for definite description resolution: A survey and a preliminary proposal. In *Proc. of the ACL Workshop on Reference Resolution*.

S. P. Ponzetto and M. Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proc. of HLT/NAACL*, pages 192–199.

W. M. Soon, H. T. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

V. Stoyanov, N. Gilbert, C. Cardie, and E. Riloff. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proc. of the ACL*.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of HLT-NAACL*, pages 252–259.

O. Uryupina. 2003. High-precision identification of discourse new and unique noun phrases. In *Proc. of the ACL Student Research Workshop*.

Y. Versley, A. Moschitti, M. Poesio, and X. Yang. 2008. Coreference systems based on kernel methods. In *Proc. of COLING*, pages 961–968.

R. Vieira and M. Poesio. 2000. Processing definite descriptions in corpora. In *Corpus-based and Computational Approaches to Discourse Anaphora*, pages 189–212. UCL Press.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proc. of MUC-6*, pages 45–52.

M. Walker, A. Joshi, and E. Prince, editors. 1998. *Centering Theory in Discourse*. Oxford University Press.

X. Yang, G. Zhou, J. Su, and C. L. Tan. 2003. Coreference resolution using competitive learning approach. In *Proc. of the ACL*, pages 176–183.

X. Yang, J. Su, G. Zhou, and C. L. Tan. 2004. An NP-cluster based approach to coreference resolution. In *Proc. of COLING*, pages 226–232.

X. Yang, J. Su, J. Lang, C. L. Tan, and S. Li. 2008. An entity-mention model for coreference resolution with inductive logic programming. In *Proc. of the ACL*, pages 843–851.

Global Learning of Noun Phrase Anaphoricity in Coreference Resolution via Label Propagation

ZHOU GuoDong KONG Fang
JiangSu Provincial Key Lab for Computer Information Processing Technology
School of Computer Science and Technology
Soochow University. Suzhou, China 215006
Email:{gdzhou,kongfang}@suda.edu.cn

Abstract

Knowledge of noun phrase anaphoricity might be profitably exploited in coreference resolution to bypass the resolution of non-anaphoric noun phrases. However, it is surprising to notice that recent attempts to incorporate automatically acquired anaphoricity information into coreference resolution have been somewhat disappointing. This paper employs a global learning method in determining the anaphoricity of noun phrases via a label propagation algorithm to improve learning-based coreference resolution. In particular, two kinds of kernels, i.e. the feature-based RBF kernel and the convolution tree kernel, are employed to compute the anaphoricity similarity between two noun phrases. Experiments on the ACE 2003 corpus demonstrate the effectiveness of our method in anaphoricity determination of noun phrases and its application in learning-based coreference resolution.

1 Introduction

Coreference resolution, the task of determining which noun phrases (NPs) in a text refer to the same real-world entity, has long been considered an important and difficult problem in natural language processing. Identifying the linguistic constraints on when two NPs can co-refer remains an active area of research in the community. One significant constraint on coreference, the anaphoricity constraint, specifies that a non-anaphoric NP cannot be coreferent with any of its preceding NPs in a given text. Therefore, it is useful to skip over these non-anaphoric NPs rather than attempt an unnecessary search for an antecedent for them, only to end up with inaccurate outcomes. Although many existing machine learning approaches to coreference resolution have performed reasonably well without explicit anaphoricity determination (e.g., Soon et al 2001; Ng and Cardie 2002b; Strube and Muller 2003; Yang et al 2003, 2008), anaphoricity determination has been studied fairly extensively in the literature, given the potential usefulness of NP anaphoricity in coreference resolution. One common approach involves the design of heuristic rules to identify specific types of non-anaphoric NPs, such as pleonastic pronouns (e.g. Paice and Husk 1987; Lappin and Leass 1994; Kennedy and Boguraev 1996; Denber 1998) and existential definite descriptions (e.g., Vieira and Poesio 2000). More recently, the problem has been tackled using statistics-based (e.g., Bean and Riloff 1999; Bergsma et al 2008) and learning-based (e.g. Evans 2001; Ng and Cardie 2002a; Ng 2004; Yang et al 2005; Denis and Balbridge 2007) methods. Although there is empirical evidence (e.g. Ng and Cardie 2002a, 2004) that coreference resolution might be further improved with proper anaphoricity information, its contribution is still somewhat disappointing and lacks systematic evaluation.

This paper employs a label propagation (LP) algorithm for global learning of NP anaphoricity. Given the labeled data and the unlabeled data, the LP algorithm first represents labeled and unlabeled instances as vertices in a connected graph, then propagates the label information from any vertex to nearby vertices through weighted edges and finally infers the labels of unlabeled instances until a global stable stage is achieved. Here, the labeled data in this paper include all the NPs in the training texts with the anaphoricity labeled and the unlabeled data include all the NPs in a test text with the anaphoricity unlabeled. One major advantage of LP-based anaphoricity determination is that the anaphoricity of all the NPs in a text can be determined together in a global way. Compared with previous methods, the LP algorithm can effectively capture the natural clustering structure in both the labeled and unlabeled data to smooth the labeling function. In particular, two kinds of

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 978–986,
Singapore, 6-7 August 2009.

kernels, i.e. the feature-based RBF kernel and the convolution tree kernel, are employed to compute the anaphoricity similarity between two NPs and weigh the edge between them. Experiments on the ACE 2003 corpus show that our LP-based anaphoricity determination significantly outperforms locally-optimized one, which adopts a classifier (e.g. SVM) to determine the anaphoricity of NPs in a text individually and significantly improves the performance of learning-based coreference resolution. It also shows that, while feature-based anaphoricity determination contributes much to pronoun resolution, its contribution on definite NP resolution can be ignored. In comparison, it shows that tree kernel-based anaphoricity resolution contributes significantly to the resolution of both pronouns and definite NPs due to the inclusion of various kinds of syntactic structured information.

The rest of this paper is organized as follows. In Section 2, we review related work in anaphoricity determination. Then, the LP algorithm is introduced in Section 3 while Section 4 describes different similarity measurements explored in the LP algorithm. Section 5 shows the experimental results. Finally, we conclude our work in Section 6.

2 Related Work

Given its potential usefulness in coreference resolution, anaphoricity determination has been studied fairly extensively in the literature and can be classified into three categories: heuristic rule-based (e.g. Paice and Husk 1987; Lappin and Leass 1994; Kennedy and Boguraev 1996; Denber 1998; Vieira and Poesio 2000), statistics-based (e.g., Bean and Riloff 1999; Cherry and Bergsma 2005; Bergsma et al 2008) and learning-based (e.g. Evans 2001; Ng and Cardie 2002a; Ng 2004; Yang et al 2005; Denis and Balbridge 2007).

For the heuristic rule-based approaches, Paice and Husk (1987), Lappin and Leass (1994), Kennedy and Boguraev (1996), Denber (1998), and Cherry and Bergsma (2005) looked for particular constructions using certain trigger words to identify pleonastic pronouns while Vieira and Poesio (2000) recognized non-anaphoric definite NPs through the use of syntactic cues and case-sensitive rules and found that nearly 50% of definite NPs are non-anaphoric. As a representative, Lappin and Leass (1994), and Kennedy and Boguraev (1996) looked for modal adjectives (e.g. "necessary") or cognitive verbs (e.g. "It is thought that …") in a set of patterned constructions.

For the statistics-based approaches, Bean and Riloff (1999) developed a statistics-based method for automatically identifying existential definite NPs which are non-anaphoric. The intuition behind is that many definite NPs are not anaphoric since their meanings can be understood from general world knowledge. They found that existential NPs account for 63% of all definite NPs and 76% of them could be identified by syntactic or lexical means. Using 1600 MUC-4 terrorism news documents as the training data, they achieved 87% in precision and 78% in recall at identifying non-anaphoric definite NPs. Cherry and Bergsma (2005) extended the work of Lappin and Leass (1994) for large-scale anaphoricity determination by additionally detecting non-anaphoric instances of *it* using Minipar's pleonastic category *Subj*. This is done by both employing Minipar's named entity recognition to identify time expressions, such as "it was midnight…", and providing a number of other linguistic patterns to match common non-anaphoric *it* cases, such as in expressions "darn it" and don't overdo it". Bergsma et al (2008) proposed a distributional method in detecting non-anaphoric pronouns by first extracting the surrounding textual context of the pronoun, then gathering the distribution of words that occurred within that context from a large corpus and finally learning to classify these distributions as representing either anaphoric and non-anaphoric pronoun instances. Experiments on the Science News corpus of It-Bank [1] in identifying non-anaphoric pronoun *it* show that their distributional method achieved the performance of 81.4%, 71.0% and 75.8 in precision, recall and F1-measure, respectively, compared with the performance of 93.4%, 21.0% and 34.3 in precision, recall and F1-measure, respectively using the rule-based approach as described in Lappin and Leass (1994), and the performance of 66.4%, 49.7% and 56.9 in precision, recall and F1-measure, respectively using the rule-based approach as described in Cherry and Bergsma (2005).

Among the learning-based methods, Evans (2001) applied a machine learning approach on identifying the non-anaphoricity of pronoun *it*. Ng and Cardie (2002a) employed various domain-independent features in identifying anaphoric NPs and showed how such information

[1] www.cs.ualberta.ca/~bergsma/ItBank/

can be incorporated into a coreference resolution system. Experiments show that their method improves the performance of coreference resolution by 2.0 and 2.6 to 65.8 and 64.2 in F1-measure on the MUC-6 and MUC-7 corpora, respectively, due to much more gain in precision compared with the loss in recall. Ng (2004) examined the representation and optimization issues in computing and using anaphoricity information to improve learning-based coreference resolution systems. He used an anaphoricity classifier as a filter for coreference resolution. Evaluation on the ACE 2003 corpus shows that, compared with a baseline coreference resolution system of no explicit anaphoricity determination, their method improves the performance by 2.8, 2.2 and 4.5 to 54.5, 64.0 and 60.8 in F1-measure (due to the gain in precision) on the NWIRE, NPAPER and BNEWS domains, respectively, via careful determination of an anaphoricity threshold with proper constraint-based representation and global optimization. However, he did not look into the contribution of anaphoricity determination on coreference resolution of different NP types, such as pronoun and definite NPs. Yang et al (2005) made use of non-anaphors to create a special class of training instances in the twin-candidate model (Yang et al 2003) and thus equipped it with the non-anaphoricity determination capability. Experiments show that the proposed method improves the performance by 2.9 and 1.6 to 67.3 and 67.2 in F1-measure on the MUC-6 and MUC-7 corpora, respectively, due to much more gain in precision compared with the loss in recall. However, surprisingly, their experiments also show that eliminating non-anaphors using an anaphoricity determination module in advance harms the performance. Denis and Balbridge (2007) employed an integer linear programming (ILP) formulation for coreference resolution which models anaphoricity and coreference as a joint task, such that each local model informs the other for final assignments. Experiments on the NWIRE, NPAPER and BNEWS domains of the ACE 2003 corpus shows that this joint anaphoricity-coreference ILP formulation improves the F1-measure by 0.7-1.0 over the coreference-only ILP formulation. However, their experiments assume true ACE mentions(i.e. all the ACE mentions are already known from the annotated corpus). Therefore, the actual effect of this joint anaphoricity-coreference ILP formulation on fully-automatic coreference resolution is still unclear.

3 Label Propagation

In the LP algorithm (Zhu and Ghahramani 2002), the natural clustering structure in data is represented as a connected graph. Given the labeled data and unlabeled data, the LP algorithm first represents labeled and unlabeled instances as vertices in a connected graph, then propagates the label information from any vertex to nearby vertices through weighted edges and finally infers the labels of unlabeled instances until a global stable stage is achieved. Figure 1 presents the label propagation algorithm.

Assume:

Y : the $n*r$ labeling matrix, where y_{ij} represents the probability of vertex $x_i (i = 1 \ldots n)$ with label $r_j (j = 1 \ldots r)$;

Y_L : the top l rows of Y^0. Y_L corresponds to the l labeled instances;

Y_U : the bottom u rows of Y^0. Y_U corresponds to the u unlabeled instances;

$\overline{T}$: a $n*n$ matrix, with $\overline{t}_{ij}$ is the probability jumping from vertex x_i to vertex x_j;

BEGIN (the algorithm)

Initialization:

1) Set the iteration index $t = 0$;
2) Let Y^0 be the initial soft labels attached to each vertex;
3) Let Y_L^0 be consistent with the labeling in the labeled data, where y_{ij}^0 = the weight of the labeled instance if x_i has the label r_j;
4) Initialize Y_U^0;

REPEAT

Propagate the labels of any vertex to nearby vertices by $Y^{t+1} = \overline{T} Y^t$;

Clamp the labeled data, that is, replace Y_L^{t+1} with Y_L^0;

UNTIL Y converges(e.g. Y_L^{t+1} converges to Y_L^0);

Assign each unlabeled instance with a label: for $x_i (l \prec i \le n)$, find its label with $\arg\max_j y_{ij}$;

END (the algorithm)

Figure 1: The LP algorithm

Here, each vertex corresponds to an instance, and the edge between any two instances x_i and x_j is weighted by w_{ij} to measure their similarity. In principle, larger edge weights allow labels to travel through easier. Thus the closer the instances are, the more likely they have similar

labels. The algorithm first calculates the weight w_{ij} using a kernel, then transforms it to $t_{ij} = p(j \to i) = w_{ij} / \sum_{k=1}^{n} w_{kj}$, which measures the probability of propagating a label from instance x_j to instance x_i, and finally normalizes t_{ij} row by row using $\bar{t}_{ij} = t_{ij} / \sum_{k=1}^{n} t_{ik}$ to maintain the class probability interpretation of the labeling matrix Y.

During the label propagation process, the label distribution of the labeled data is clamped in each loop using their initial weights and acts like forces to push out labels through the unlabeled data. With this push originating from the labeled data, the label boundaries will be pushed faster along edges with larger weights and settle in gaps along those with lower weights. Ideally, we can expect that w_{ij} across different classes should be as small as possible and w_{ij} within the same class as big as possible. In this way, label propagation tends to happen within the same class. This algorithm has been shown to converge to a unique solution (Zhu and Ghahramani 2002), which can be obtained without iteration in theory, and the initialization of Y_U^0 (the unlabeled data) is not important since Y_U^0 does not affect its estimation. However, proper initialization of Y_U^0 actually helps the algorithm converge more rapidly in practice. In this paper, each row in Y_U^0, i.e. the label distribution for each test instance, is initialized to the weighted similarity of the test instance with the labeled instances.

4 Kernel-based Similarity

The key issue in label propagation is how to compute the similarity w_{ij} between two instances x_i and x_j. This paper examines two similarity measures: the feature-based RBF kernel and the convolution tree kernel.

Feature Type	Feature	Description
Features related with current NP itself	IsPronoun	1 if current NP is a pronoun, else 0
	IsDefiniteNP	1 if current NP is a define NP, else 0
	IsDemonstrativeNP	1 if current NP is a demonstrative NP, else 0
	IsArg0	1 if the semantic role of current NP is Arg0/agent, else 0
	IsArg0MainVerb	1 if current NP has the semantic role of Arg0/agent for the main predicate of the sentence, else 0
	IsArgs	0 if current NP has no semantic role, else 1
	IsSingularNP	1 if current NP is a singular noun, else 0
	IsMaleFemalePronoun	1 if current NP is a male/female personal pronoun, else 0
Features related with the local context surrounding current NP	StringMatch	1 if there is a full string match between current NP and one of other phrases in the context, else 0
	NameAlias	1 if current NP and one of other phrases in the context is a name alias or abbreviation of the other, else 0
	Appositive	1 if current NP and one of other phrases in the context are in an appositive structure, else 0
	NPNested	1 if current NP is nested in another NP, else 0
	NPNesting	1 if current NP nests another NP, else 0
	WordSenseAgreement	1 if current NP and one of other phrases in the context agree in the WordNet sense, else 0
	IsFirstNPinSentence	1 if current NP is the first NP of this sentence, else 0
	BackwardDistance	The distance between current NP and the nearest backward clause, indicated by coordinating words (e.g. that,which).
	ForwardDistance	The distance between the nearest forward clause, indicated by coordinating words (e.g. that, which), and current NP.

Table 1: Features in anaphoricity determination of NPs. Note: the semantic role-related features are derived from an in-house state-of-the-art semantic role labeling system.

4.1 Feature-based Kernel

In our feature-based RBF kernel to anaphoricity determination, an instance is represented by 17 lexical, syntactic and semantic features, as shown in Table 1, which are specifically designed for distinguishing anaphoric and non-anaphoric NPs, according to common-sense knowledge and linguistic intuitions. Since the local context surrounding an NP plays a critical role in discriminating whether an NP is anaphoric or not, the features in Table 1 can be classified into two categories: (a) current NP (i.e. the NP in anaphoricity consideration) itself, e.g.

types and semantic roles of current NP; (b) contextual information, e.g. whether current NP is nested in another NP, the distance between current NP and a clause structure, indicated by coordinating words (e.g. that, this, which).

4.2 Tree Kernel

Given a NP in anaphoricity determination, a parse tree represents the local context surrounding current NP in a structural way and thus contains much information in determining whether current NP is anaphoric or not. For example, the commonly used knowledge for anaphoricity determination, such as the grammatical role of current NP or whether current NP is nested in other NPs, can be directly captured by a parse tree structure.

Given a parse tree and a NP in consideration, the problem is how to choose a proper parse tree structure to cover syntactic structured information well in the tree kernel computation. Generally, the more a parse tree structure includes, the more syntactic structured information would be provided, at the expense of more noisy/unnecessary information. In this paper, we limit the window size to 5 chunks (either NPs or non-NPs), including previous two chunks, current chunk (i.e. current NP) and following two chunks, and prune out the substructures outside the window. Figure 2 shows the full parse tree for the sentence "Mary said the woman in the room hit her too", using the Charniak parser (Charniak 2001), and the chunk sequence derived from the parse tree using the Perl script[2] written by Sabine Buchholz from Tilburg University.

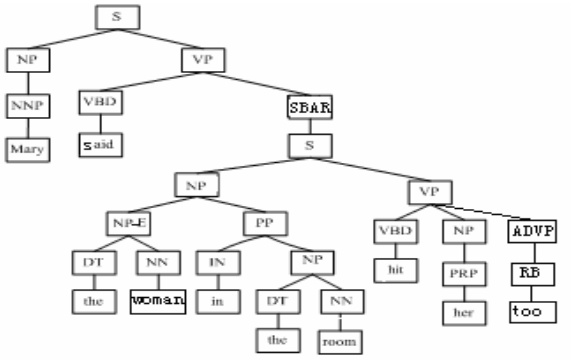
Here, we explore four parse tree structures in NP anaphoricity determination: the common tree (CT), the shortest path-enclosed tree (SPT), the minimum tree (MT) and the dynamically extended tree (DET), motivated by Yang et al (2006) and Zhou et al (2008). Following are the examples of the four parse tree structures, corresponding to the full parse tree and the chunk sequence, as shown in Figure 2, with the NP chunk "(NP (DT the) (NN woman))" in anaphoricity determination.

Common Tree (CT)

As shown in Figure 3(a), CT is the complete sub-tree rooted by the nearest common ancestor of the first chunk "(NP (NNP Mary))" and the last chunk "(NP (DT the) (NN room))" of the five-chunk window.

Shortest Path-enclosed Tree (SPT)

As shown in Figure 3(b), SPT is the smallest common sub-tree enclosed by the shortest path between the first chunk "(NP (NNP Mary))" and the last chunk "(NP (DT the) (NN room))" of the five-chunk window.

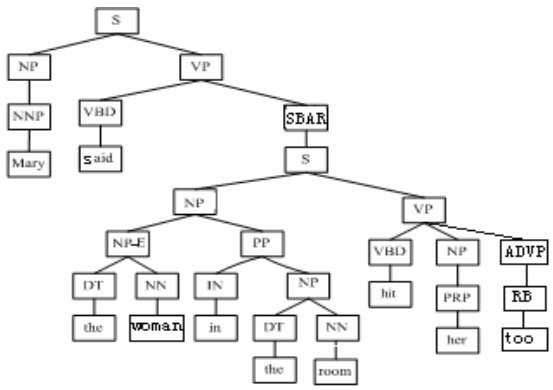


(a) the full parse tree

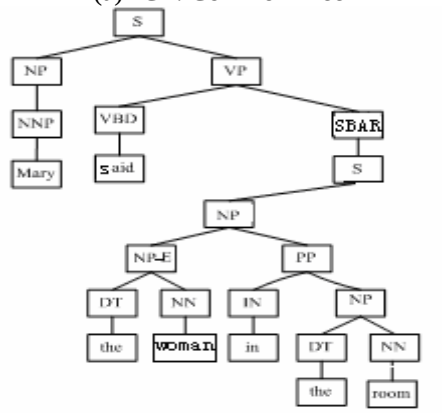
(NP (NNP Mary)) (VP (VBD said)) ***(NP-E (DT the) (NN woman))*** (PP (IN in)) (NP (DT the) (NN room)) (VP (VBD hit)) (NP (PRP her)) (ADVP (RB too))

(b) the chunk sequence

Figure 2: The full parse tree for the sentence "Mary said the woman in the room hit her too", using the Charniak parser, and the corresponding chunk sequence derived from it. Here, the label "E" indicates the NP in consideration.

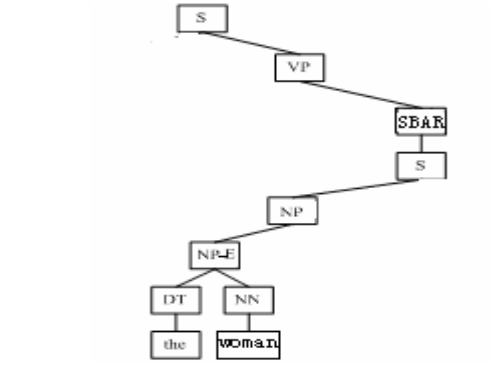


(a) CT: Common Tree

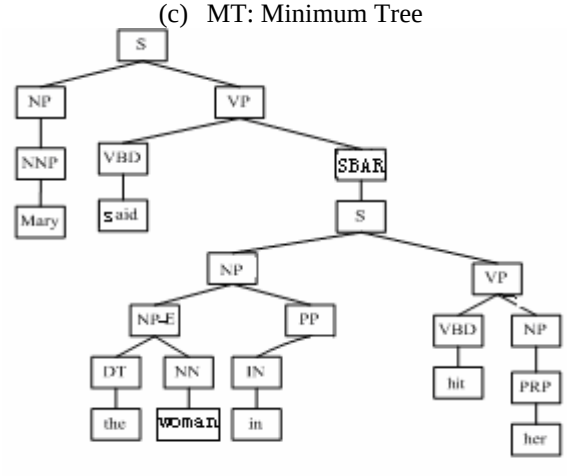


(b) SPT: Shortest Path-enclosed Tree

[2] http://ilk.kub.nl/~sabine/chunklink/



(c) MT: Minimum Tree



(d) DET: Dynamically Extended Tree

Figure 3: Examples of parse tree structures.

Minimum Tree (MT)

As shown in Figure 3(c), MT only keeps the root path from the NP in anaphoricity determination to the root node of SPT.

Dynamically Extended Tree (DET),

The intuitions behind DET are that the information related with antecedent candidates (all the antecedent candidates compatible[3] with current NP in anaphoricity consideration), predicates[4] and right siblings plays a critical role in coreference resolution. Given a MT, this is done by:

1) Attaching all the compatible antecedent candidates and their corresponding paths. As shown in Figure 3(d), "Mary" is attached while "the room" is not since the former is compatible with the NP "the woman" and the latter is not compatible with the NP "the woman". In this way, possible coreference between current NP and the compatible antecedent candidates can be included in the parse tree structure. In some sense, this is a natural extension of the twin-candidate learning method proposed in Yang et al (2003), which explicitly models the competition between two antecedent candidates.
2) For each node in MT, attaching the path from the node to the leaf node of the corresponding predicate, if it is predicate-headed, in the sense that such predicate-related information is useful in identifying certain kinds of expressions with non-anaphoric NPs, e.g. the non-anaphoric *it* in "darn it". As shown in Figure 3(d), "said" and "hit" are attached.
3) Attaching the path to the head word of the first right sibling if the parent of current NP is a NP and current NP has one or more right siblings. Normally, the NP in anaphoricity consideration, NP-E, in the production of "NP->NP-E+PP" introduces a new entity and thus non-anaphoric.
4) Pruning those nodes (except POS nodes) with the single in-arc and the single out-arc and with its syntactic phrase type same as its child node.

In this paper, the similarity between two parse trees is measured using a convolution tree kernel, which counts the number of common sub-trees as the syntactic structure similarity between two parse trees. For details, please refer to Collins and Duffy (2001).

5 Experimentation

We have systematically evaluated the label propagation algorithm on global learning of NP anaphoricity determination on the ACE 2003 corpus, and its application in coreference resolution.

5.1 Experimental Setting

The ACE 2003 corpus contains three domains: newswire (NWIRE), newspaper (NPAPER), and broadcast news (BNEWS). For each domain, there exist two data sets, training and devtest, which are used for training and testing respectively.

As a baseline coreference resolution system, a raw test text is first preprocessed automatically by a pipeline of NLP components, including sentence boundary detection, POS tagging, named entity recognition and phrase chunking, and then a training or test instance is formed by a anaphor and one of its antecedent candidates, similar to Soon et al (2001). Among them, named entity recognition, part-of-speech tagging and noun phrase chunking apply the same Hidden Markov Model (HMM)-based engine with

[3] With matched number, person and gender agreements.

[4] For simplicity, only verbal predicates are considered in this paper. However, this can be extended to nominal predicates with automatic identification of nominal predicates.

error-driven learning capability (Zhou and Su, 2000 & 2002). During training, for each anaphor encountered, a positive instance is created by pairing the anaphor and its closest antecedent while a set of negative instances is formed by pairing the anaphor with each of the non-coreferential candidates. Based on the training instances, a binary classifier is generated using a particular learning algorithm. In this paper, we use SVMLight developed by Joachims (1998). During resolution, an anaphor is first paired in turn with each preceding antecedent candidate to form a test instance, which is presented to a classifier. The classifier then returns a confidence value indicating the likelihood that the candidate is the antecedent. Finally, the candidate with the highest confidence value is selected as the antecedent. As a baseline, the NPs with mismatched number, person and gender agreements are filtered out. On average, an anaphor has ~7 antecedent candidates. In particular, the test corpus is resolved in document-level, i.e. one document by one document.

For anaphoricity determination, we report the performance in Acc^+ and Acc^-, which measure the accuracies of identifying anaphoric NPs and non-anaphoric NPs, respectively. Obviously, higher Acc^+ means that more anaphoric NPs would be identified correctly, while higher Acc^- means that more non-anaphoric NPs would be filtered out. For coreference resolution, we report the performance in terms of recall, precision, and F1-measure using the commonly-used model theoretic MUC scoring program (Vilain et al., 1995). For separate scoring of different NP types, a recognized reference is considered correct if the reconized antecedent is in the coreferential chain of the anaphor. To see whether an improvement is significant, we conduct significance testing using paired t-test. In this paper, '>>>', '>>' and '>' denote p-values of an improvement smaller than 0.01, in-between (0.01, 0,05] and bigger than 0.05, which mean significantly better, moderately better and slightly better, respectively.

5.2 Experimental Results

Table 2 shows the performance of LP-based anaphoricity determination using the feature-based RBF kernel. It shows that our method achieves the accuracies of 74.8/84.4, 76.2/81.3 and 71.8/81.7 on identifying anaphoric/non-anaphoric NPs in the NWIRE, NPAPER and BNEWS domains, respectively. This suggests that our approach can effectively filter out about 82% of non-anaphoric NPs. However, it can only keep about 74% of anaphoric NPs. Table 2 also shows the performance on different NP types. Considering the effectiveness of anaphoricity determination on indefinite NPs (due to that most of anaphoric indefinite NPs are in an appositive structure and thus can be easily captured by the IsAppositive feature) and that most of errors in anaphoricity determination on proper nouns are caused by the named entity recognition module in the preprocessing), it indicates the difficulty of anaphoricity determination in filtering out non-anaphoric pronouns and identifying anaphoric definite NPs. As a comparison, Table 2 also shows the performance of locally-optimized anaphoricity determination using a classifier (SVM with the feature-based RBF kernel, as adopted in this paper) to determine the NPs in a text individually. It shows that the LP-based method systematically outperforms (>>>) the SVM-based method. This suggests the effectiveness of the LP algorithm in global modeling of the natural clustering structure in anaphoricity determination.

Table 3 shows the performance of LP-based anaphoricity determination using the convolution tree kernel on different parse tree structures. It shows that while MT performed worst due to its simple structure, DET outperforms MT(>>>), SPT(>>>) and CT(>>>) on all the three domains due to fine inclusion of necessary structural information, although inclusion of more information in both CT and SPT also improves the performance. It again verifies that LP-based anaphoricity determination outperforms (>>>) SVM-based one, using the tree kernel. Table 4 further indicates that all the three kinds of structural information related with antecedent candidates, predicates and right siblings in DET contribute significantly (>>>). In addition, Table 5 shows the detailed performance of LP-based anaphoricity determination on different anaphor types using DET. Compared with the feature-based RBF kernel as shown in Table 2, it shows that the convolution tree kernel significantly outperforms (>>>) the feature-based RBF kernel in all the three domains, with much contribution due to performance improvement on both pronouns and definite NPs, although the tree kernel performs moderately worse than the feature-based RBF kernel due to the effectiveness of anaphoricity determination on proper nouns and indefinite NPs using the IsNameAlias and IsAppositive features respectively.

Anaphor Type	NWIRE		NPAPER		BNEWS	
	Acc^+ (%)	Acc^- (%)	Acc^+ (%)	Acc^- (%)	Acc^+ (%)	Acc^- (%)
Pronoun	88.7	56.2	90.2	58.6	87.4	57.8
ProperNoun	72.5	85.2	74.6	80.5	70.6	78.8
DefiniteNP	66.6	83.1	72.1	77.5	65.3	81.5
InDefiniteNP	95.4	93.7	90.5	95.8	87.2	97.3
Overall	74.8	84.4	76.2	81.3	71.8	81.7
Overall(SVM)	*71.3*	*80.2*	*73.5*	*79.1*	*68.4*	*78.6*

Table 2: The performance of LP-based anaphoricity determination using the feature-based RBF kernel

Parse Tree structure Scheme		NWIRE (%)	NPAPER (%)	BNEWS (%)
CT	Acc^+	72.6	74.3	74.2
	Acc^-	82.1	80.2	72.3
SPT	Acc^+	72.4	74.1	73.8
	Acc^-	80.8	79.5	72.5
MT	Acc^+	71.4	70.5	66.9
	Acc^-	77.2	75.3	78.2
DET	Acc^+	79.2	81.2	76.5
	Acc^-	87.8	84.5	85.3
DET(SVM)	*Acc^+*	*76.5*	*78.9*	*74.3*
	Acc^-	*82.3*	*81.6*	*83.2*

Table 3: The performance of LP-based anaphoricity determination using the convolution tree kernel on different parse tree structures

Performance Change		NWIRE (%)	NPAPER (%)	BNEWS (%)
- antecedent candidates	Acc^+	-4.0	-3.8	-4.3
	Acc^-	-5.2	-5.3	-4.5
-predicate	Acc^+	-5.2	-4.8	-5.6
	Acc^-	-4.3	-3.5	-4.9
-first right sibling	Acc^+	-3.6	-4.1	-3.1
	Acc^-	-4.8	-5.2	-4.4

Table 4: The contribution of structural information in DET

Anaphor Type	NWIRE		NPAPER		BNEWS	
	Acc^+ (%)	Acc^- (%)	Acc^+ (%)	Acc^- (%)	Acc^+ (%)	Acc^- (%)
Pronoun	90.1	75.6	90.7	79.2	89.2	77.5
ProperNoun	71.4	83.5	72.8	78.1	68.3	77.2
DefiniteNP	74.6	89.1	77.3	85.5	75.3	88.7
InDefiniteNP	93.2	92.1	90.2	94.2	89.4	95.5
Overall	79.2	87.8	81.2	84.5	76.5	85.3

Table 5: The performance of LP-based anaphoricity determination using the tree kernel on DET

Finally, we evaluate the effect of LP-based anaphoricity determination on coreference resolution by including it as a preprocessing step to a baseline coreference resolution system without explicit anaphoricity determination, which employs the same set of features, as adopted in the single-candidate model of Yang et al (2003), using a SVM-based classifier and the feature-based RBF kernel. It shows that anaphoricity determination with the feature-based RBF Kernel much improves (>>>) the performance of coreference resolution with most of the contribution due to pronoun resolution while its contribution on definite NPs can be ignored. It indicates the usefulness of anaphoricity determination in filtering out non-anaphoric pronouns and the difficulty in identifying anaphoric definite NPs, using the feature-based RBF kernel. It also shows that tree kernel-based anaphoricity determination can not only improve (>>>) the performance on pronoun resolution but also improve (>>>) the performance on definite NP resolution due to the much better performance of tree kernel-based anaphoricity determination on definite NPs. This suggests the necessity of exploring structural information in identifying anaphoric definite NPs.

System		NWIRE			NPAPER			BNEWS		
		R%	P%	F	R%	P%	F	R%	P%	F
BaseLine (No Anaphoricity)	Pronoun	66.5	61.6	64.0	70.1	64.2	67.0	61.7	63.2	62.4
	DefiniteNP	26.9	80.3	40.2	34.5	62.4	44.4	30.5	71.4	42.9
	Overall	53.1	67.4	59.4	57.7	67.0	62.1	48.0	65.9	55.5
+Anaphoricity determination with the feature-based RBF kernel	Pronoun	64.1	67.9	66.0	67.3	72.4	69.8	59.5	75.7	66.6
	DefiniteNP	26.7	80.6	40.3	34.2	62.5	44.3	30.4	71.9	43.1
	Overall	50.6	75.4	60.7	54.4	77.1	63.8	45.9	76.9	57.4
+Anaphoricity determination with the convolution tree kernel	Pronoun	63.5	70.9	67.0	68	74.9	71.3	61.1	77.6	68.3
	DefiniteNP	28.5	82.4	42.1	36.2	65.3	46.1	32.3	73.1	44.2
	Overall	51.6	77.2	61.8	55.2	78.6	65.2	47.5	80.3	59.6

Table 6: Employment of anaphoricity determination in coreference resolution

6 Conclusion

This paper systematically studies a global learning method in identifying the anaphoricity of noun phrases via a label propagation algorithm and the application of an explicit anaphoricity determination module in improving learning-based coreference resolution. In particular, two kinds of kernels, i.e. the feature-based RBF kernel and the convolution tree kernel, are employed to compute the anaphoricity similarity

between two NPs. Evaluation on the ACE 2003 corpus indicates that LP-based anaphoricity determination using both the kernels much improves the performance of coreference resolution. It also shows the usefulness of various structural information, related with antecedent candidates, predicates and right siblings, in tree kernel-based anaphoricity determination and in coreference resolution of both pronouns and definite NPs.

To our knowledge, this is the first systematic exploration of both feature-based and tree kernel methods in anaphoricity determination and the application of an explicit anaphoricity determination module in learning coreference resolution.

Acknowledgement

This research is supported by Project 60873150 under the National Natural Science Foundation of China, project 2006AA01Z147 under the "863" National High-Tech Research and Development of China, project 200802850006 under the National Research Foundation for the Doctoral Program of Higher Education of China.

References

Bean D. and Riloff E. (1999). Corpus-based Identification of Non-Anaphoric Noun Phrases. *ACL'1999*:373-380.

Bergsma S., Lin D.K. and Goebel R. (2008). Distributional Identification of Non-Referential Pronouns. *ACL'2008*: 10-18.

Charniak E. (2001). Immediate-head Parsing for Language Models. *ACL'2001*: 129-137.

Cherry C. and Bergsma S. (2005). An expectation maximization approach to pronoun resolution. *CoNLL'2005*:88-95.

Collins M. and Duffy N. (2001). Convolution kernels for natural language. *NIPS'2001*: 625-632.

Denber M. (1998). Automatic Resolution of Anaphora in English. Technical Report, Eastman Kodak Co.

Denis P. and Baldridge J. (2007). Joint determination of anaphoricity and coreference using integer programming. *NAACL-HLT'2007*:236-243.

Evans R. (2001). Applying machine learning toward an automatic classification of *it*. *Literary and Linguistic Computing*, 16(1):45.57.

Joachims T. (1998). Text Categorization with Support Vector Machine: learning with many relevant features. *ECML-1998*: 137-142.

Kennedy C. and Boguraev B. (1996). Anaphora for everyone: pronominal anaphora resolution without a parser. *COLING'1996*: 113-118.

Lappin S. and Leass H.J. (1994). An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535.561.

Ng V. and Cardie C. (2002a). Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. *COLING'2002*:730-736.

Ng V. and Cardie C. (2002b). Improving machine learning approaches to coreference resolution. *ACL'2002*: 104-111

Ng V. (2004). Learning Noun Phrase Anaphoricity to Improve Conference Resolution: Issues in Representation and Optimization. *ACL'2004*: 151-158

Paice C.D. and Husk G.D. (1987). Towards the automatic recognition of anaphoric features in English text: the impersonal pronoun *it*. *Computer Speech and Language*, 2:109-132.

Soon W.M., Ng H.T. and Lim D. (2001). A machine learning approach to coreference resolution of noun phrase. *Computational Linguistics*, 2001, 27(4):521-544.

Strube M. and Muller C. (2003). A machine learning approach to pronoun resolution in spoken dialogue. *ACL'2003*: 168-175

Vieira R. and Poesio M. (2000). An empirically based system for processing definite descriptions. *Computational Linguistics*, 27(4): 539–592.

Vilain M., Burger J., Aberdeen J., Connolly D. and Hirschman L. (1995). A model theoretic coreference scoring scheme. *MUC-6*: 45–52.

Yang X.F., Zhou G.D., Su J. and Chew C.L. (2003). Coreference Resolution Using Competition Learning Approach. *ACL'2003*:177-184

Yang X.F., Su J. and Tan C.L. (2005). A Twin-Candidate Model of Coreference Resolution with Non-Anaphor Identification Capability. *IJCNLP'2005*:719-730.

Yang X.F., Su J. and Tan C.L. (2006). Kernel-based pronoun resolution with structured syntactic knowledge. *COLING-ACL'2006*: 41-48.

Yang X.F., Su J., Lang J., Tan C.L., Liu T. and Li S. (2008). An Entity-Mention Model for Coreference Resolution with Inductive Logic Programming. *ACL'2008*: 843-851.

Zhou G.D. and Su. J. (2000). Error-driven HMM-based chunk tagger with context-dependent lexicon. *EMNLP-VLC'2000:* 71–79

Zhou G.D. and Su J. (2002). Named Entity recognition using a HMM-based chunk tagger. In *ACL'2002*:473–480.

Zhou G.D., Kong F. and Zhu Q.M. (2008). Context-sensitive convolution tree kernel for pronoun resolution . *IJCNLP'2008*:25-31.

Zhu X. and Ghahramani Z. (2002). Learning from Labeled and Unlabeled Data with Label Propagation. *CMU CALD Technical Report*.CMU-CALD-02-107.

Employing the Centering Theory in Pronoun Resolution from the Semantic Perspective

KONG Fang ZHOU GuoDong* ZHU Qiaoming
JiangSu Provincial Key Lab for Computer Information Processing Technology
School of Computer Science and Technology
Soochow University. Suzhou, China 215006
Email: {kongfang, gdzhou, qmzhu}@suda.edu.cn

Abstract

In this paper, we employ the centering theory in pronoun resolution from the semantic perspective. First, diverse semantic role features with regard to different predicates in a sentence are explored. Moreover, given a pronominal anaphor, its relative ranking among all the pronouns in a sentence, according to relevant semantic role information and its surface position, is incorporated. In particular, the use of both the semantic role features and the relative pronominal ranking feature in pronoun resolution is guided by extending the centering theory from the grammatical level to the semantic level in tracking the local discourse focus. Finally, detailed pronominal subcategory features are incorporated to enhance the discriminative power of both the semantic role features and the relative pronominal ranking feature. Experimental results on the ACE 2003 corpus show that the centering-motivated features contribute much to pronoun resolution.

1 Introduction

Coreference accounts for cohesion in a text and is, in a sense, the hyperlink for a natural language. Especially, a coreference instance denotes an identity of reference and holds between two referring expressions, which can be named entities, definite noun phrases, pronouns and so on. Coreference resolution is the process of linking together multiple referring expressions of a given entity in the world. The key in coreference resolution is to determine the antecedent for each referring expression in a text. The ability of linking referring expressions both within a sentence and across the sentences in a text is critical to discourse and language understanding in general. For example, coreference resolution is a key task in information extraction, machine translation, text summarization, and question answering.

There is a long tradition of research on coreference resolution within computational linguistics. While earlier knowledge-lean approaches heavily depend on domain and linguistic knowledge (Carter 1987; Carbonell and Brown 1988) and have significantly influenced the research, the later approaches usually rely on diverse lexical, syntactic and semantic properties of referring expressions (Soon et al., 2001;Ng and Cardie, 2002; Zhou et al., 2004). Current research has been focusing on exploiting semantic information in coreference resolution. For example, Yang et al (2005) proposed a template-based statistical approach to compute the semantic compatibility between a pronominal anaphor and an antecedent candidate, and Yang and Su (2007) explored semantic relatedness information from automatically discovered patterns, while Ng (2007) automatically induced semantic class knowledge from a treebank and explored its application in coreference resolution.

Particularly, this paper focuses on the centering theory (Sidner,1981;Grosz et al.,1995; Tetreault,2001), which reveals the significant impact of the local focus on referring expressions in that the antecedent of a referring expression usually depends on the center of attention throughout the local discourse segment (Mitkov,1998). Although the centering theory has been considered as a critical theory and the driving force behind the coreferential phenomena since its proposal, its application in coreference resolution (in particular pronoun resolution) has been somewhat disappointing: it fails to improve or even harms the performance of the state-of-

* Corresponding author

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 987–996,
Singapore, 6-7 August 2009.

the-art coreference resolution systems in previous research (e.g. Yang et al. 2004). This may be due to that centering was originally proposed as a model of discourse coherence instead of coreference.

The purpose of this paper is to employ the centering theory in pronoun resolution by extending it from the grammatical level to the semantic level. The intuition behind our approach is that, via determining the semantic roles of referring expressions in a sentence, such as agent and patient, we can derive various centering theory-motivated features in tracking the continuity or shift of the local discourse focus, thus allowing us to include document-level event descriptive information in resolving the coreferential relations between referring expressions.

To the best of our knowledge, this is the first research, which successfully applies the centering theory in pronoun resolution from the semantic perspective.

The rest of this paper is organized as follows. Section 2 briefly describes related work in employing the centering theory and semantic information in coreference resolution. Then, the centering theory is introduced in Section 3 while Section 4 details how to employ the centering theory from the semantic perspective. Section 5 reports and discusses the experimental results. Finally, we conclude our work in Section 6.

2 Related Work

This section briefly overviews the related work in coreference resolution from both the centering theory and semantic perspectives.

2.1 Centering Theory

In the literature, there has been much research in the centering theory and its application to coreference resolution.

In the centering theory itself, since the original work of Sidner (1979) on immediate focusing of pronouns and the subsequent work of Joshi and Weinstein (1981) on centering and inferences, much research has been done, including centering and linguistic realizations (Cote 1993; Prince and Walker 1995), empirical and psycholinguistic evaluation of centering predictions (Gordon et al 1993,1995; Brennan 1995; Walker et al 1998; Kibble 2001), and the cross-linguistic work on centering (Ziv and Crosz1994).

In applications of the centering theory to coreference resolution, representative work includes Brennan et al. (1987), Strube (1998), Tetreault (1999) and Yang et al. (2004). Brennan et al. (1987) presented a centering theory-based formalism in modeling the local focus structure in discourse and used it to track the discourse context in binding occurring pronouns to corresponding entities. In particular, a BFP (Brennan, Friedman and Pollard) algorithm is proposed to extend the original centering model to include two additional transitions called smooth shift and rough shift. Strube (1998) proposed an S-list model, assuming that a referring expression prefers a hearer-old discourse entity to other hearer-new candidates. Tetreault (1999) further advanced the BFP algorithm by adopting a left-to-right breadth first walk of the syntactic parse trees to rank the antecedent candidates. However, the above methods have not been systematically evaluated on large annotated corpora, such as MUC and ACE. Thus their effects are still unclear in real coreference resolution tasks. Yang et al (2004) presented a learning-based approach by incorporating several S-list model-based features to improve the performance in pronoun resolution. It shows that, although including S-list model-based features can slightly boost the performance in the ideal case (i.e. given the correct antecedents of anaphor's candidates), it deteriorates the overall performance in F-measure with slightly higher precision but much lower recall, in real cases, where the antecedents of anaphor's candidates are determined automatically by a separate coreference resolution module.

2.2 Semantic Information

It is well known that semantic information plays a critical role in coreference resolution. Besides the common practice of employing a thesaurus (e.g. WordNet) in semantic consistency checking, much research has been done to explore various kinds of semantic information, such as semantic similarity (Harabagiu et al 2000), semantic compatibility (Yang et al 2005, 2007), and semantic class information (Soon et al 2001; Ng 2007). Although these methods have been proven useful in coreference resolution, their contributions are much limited. For example, Ng (2007) showed that semantic similarity information and semantic agreement information could only improve the performance of coreference resolution by 0.6 and 0.5 in F-measure respectively, on the ACE 2003 NWIRE corpus.

3 Centering Theory

The centering theory is a theory about the local discourse structure that models the interaction of referential continuity and the salience of discourse entities in the internal organization of a text. In natural languages, a given entity may be referred by different expressions and act as different grammatical roles throughout a text. For example, people often use pronouns to refer to the main subject of the discourse in focus, which can change over different portions of the discourse. One main goal of the centering theory is to track the focus entities throughout a text.

The main claims of the centering theory can be formalized in terms of C_b (the backward-looking center), C_f (a list of forward-looking centers for each utterance U_n) and C_p (the preferred center, i.e. the most salient candidate for subsequent utterances). Given following two sentences: 1) Susan^i gave Betsy^j a pet hamster^k; 2) She$_i$ reminded her$_j$ that such hamsters$_k$ were quite shy. We can have U_b, U_f and U_p as follows: U_b= "Susan"; U_f={"Susan", "Betsy", "a pet hamster"}; U_p= "Susan".

	$C_b(U_n)=C_b(U_{n-1})$ or $C_b(U_{n-1})$ undefined	$C_b(U_n)\neq C_b(U_{n-1})$
$C_b(U_n)=C_p(U_n)$	Continue	Smooth Shift
$C_b(U_n)\neq C_p(U_n)$	Retain	Rough Shift

Table 1: Transitions in the centering theory

Constraints
C1. There is precisely one C_b.
C2. Every element of $C_f(U_n)$ must be realized in U_n.
C3. $C_b(U_n)$ is the highest-ranked element of $C_f(U_{n-1})$ that is realized in U_n.
Rules
R1. If some element of $C_f(U_{n-1})$ is realized as a pronoun in U_n, then so is $C_b(U_n)$.
R2. Transitions have the descending preference order of "Continue > Retain > Smooth Shift > Rough Shift".

Table 2: Constraints and rules in the centering theory

Furthermore, several kinds of focus transitions are defined in terms of two tests: whether C_b stays the same (i.e. $C_b(U_{n+1})=C_b(U_n)$), and whether C_b is realized as the most prominent referring expression (i.e. $C_b(U_n=C_p(U_n))$. We refer to the first test as cohesion, and the second test as salience. Therefore, there are four possible combinations, which are displayed in Table 1 and can result in four kinds of transitions, namely Continue, Retain, Smooth Shift, and Rough Shift. Obviously, salience, which chooses a proper verb form to make C_b prominent within a clause or sentence, is an important matter for sentence planning, while cohesion, which orders propositions in a text to maintain referential continuity, is an important matter for text planning.

Finally, the centering theory imposes several constraints and rules over C_b/C_f and above transitions, as shown in Table 2.

Given the centering theory as described above, we can draw the following conclusions:

1) The centering theory is discourse-related and centers are discourse constructs.
2) The backward-looking center C_b of U_n depends only on the expressions that constitute the utterance. That is, it is independent of its surface position and grammatical roles. Moreover, it is not constrained by any previous utterance in the segment. While the elements of $C_f(U_n)$ are partially ordered to reflect relative prominence in U_n, grammatical role information is often a major determinant in ranking C_f, e.g. in the descending priority order of "Subject > Object > Others" in English (Grosz and Joshi, 2001).
3) Psychological research (Gordon et al. 1993) and cross-linguistic research (Kameyama 1986, 1988; Walker et al. 1990,1994) have validated that C_b is preferentially realized by a pronoun in English.
4) Frequent rough shifts would lead to a lack of local cohesion. To keep local cohesion, people tend to plan ahead and minimize the number of focus shifts.

In this paper, we extend the centering theory from the grammatical level to the semantic level in attempt to better model the continuity or shift in the local discourse focus and improve the performance of pronoun resolution via centering-motivated semantic role features.

4 Employing Centering Theory from Semantic Perspective

In this section, we discuss how to employ the centering theory in pronoun resolution from the semantic perspective. In Subsection 4.1, we introduce the semantic roles. In Subsection 4.2, we introduce how to employ the centering theory in pronoun resolution via semantic role features. Finally we compare our method with the previous work in Subsection 4.3.

4.1 Semantic Role

A semantic role is the underlying relationship that a participant has with a given predicate in a clause, i.e. the actual role a participant plays in

an event, apart from linguistic encoding of the situation. If, in some situation, someone named "John" purposely hits someone named "Bill", then "John" is the agent and "Bill" is the patient of the hitting event. Therefore, given the predicate "hit" in both of the following sentences, "John" has the same semantic role of agent and "Bill" has the same semantic role of patient: 1) John hit Bill. 2) Bill was hit by John.

In the literature, labeling of such semantic roles has been well defined by the SRL (Semantic Role Labeling) task, which first identifies the arguments of a given predicate and then assigns them appropriate semantic roles. During the last few years, there has been growing interest in SRL. For example, CoNLL 2004 and 2005 have made this problem a well-known shared task. However, there is still little consensus in the linguistic and NLP communities about what set of semantic role labels are most appropriate. Typical semantic roles include core roles, such as agent, patient, instrument, and adjunct roles (such as locative, temporal, manner, and cause). For core roles, only agent and patient are consistently defined across different predicates, e.g. in the popular PropBank (Palmer et al. 2005) and the derived version evaluated in the CoNLL 2004 and 2005 shared tasks, as ARG0 and ARG1.

In this paper, we extend the centering theory from the grammatical level to the semantic level for its better application in pronoun resolution via proper semantic role features due to three reasons:

Sentence	Grammatical Role	Semantic Role
Bob opened the door with a key.	Bob: SUBJECT	Bob: AGENT
The key opened the door.	The key: SUBJECT	The key : INSTRUMENT
The door opened.	The door: SUBJECT	The door: PATIENT

Table 3: Relationship between grammatical roles and semantic roles: an example

1) Semantic roles are conceptual notions, whereas grammatical roles are morph-syntactic. While the original centering theory mainly builds from the grammatical perspective and grammatical roles do not always correspond directly to semantic roles (Table 3 shows an example of various semantic roles which a subject can play), there is a close relationship between semantic roles and grammatical roles. The statistics in the CoNLL 2004 and 2005 shared tasks (Shen and Lapata, 2007) shows that the semantic roles of ARG0/agent and ARG1/patient account for 85% of all arguments and most likely act as the centers of the local focus structure in discourse due to the close relationship between subject/object and agent/patient. Therefore, it is appropriate to model the centers of an utterance from the semantic perspective via semantic roles.
2) In a sense, semantic roles imply the information of grammatical roles, especially for subject/object. For example, the position of an argument and the voice of the predicate verb play a central role in SRL. In intuition, an argument, which occurs before an active verb and has the semantic role of Arg0/agent, tends to be a subject. That is to say, semantic roles (e.g. Arg0/agent and Arg1/patient) can be mapped into their corresponding grammatical roles (e.g. subject and object), using some heuristic rules. Therefore, it would be interesting to represent the centers of the utterances and employ the centering theory from the semantic perspective.
3) Semantic role labeling has been well studied in the literature and there are good ready-to-use toolkits available. For example, Pradhan (2005) achieved 82.2 in F-measure on the CoNLL 2005 version of the Propbank. In contrast, the research on grammatical role labeling is much less with the much lower state-of-the-art performance of 71.2 in F-measure (Buchholz, 1999). Therefore, it may be better to explore the centering theory from the semantic perspective.

4.2 Designing Centering-motivated Features from Semantic Perspective

In this paper, the centering theory is employed in pronoun resolution via three kinds of centering-motivated features:

1) Semantic role features. They are achieved by checking possible semantic roles of referring expressions with regard to various predicates in a sentence. Due to the close relationship between subject/object and agent/patient, semantic role information should be also a major determinant in deciding the center of an utterance, which is likely to be the antecedent of a referring expression in the descending priority order of "Agent > Patient > Others" with regard to their semantic roles, corresponding to the descending priority order of "Subject > Object > Others" with regard to their grammatical roles.

2) Relative pronominal ranking feature. Due to the predominance of pronouns in tracking the local discourse structure[1], the relative ranking of a pronoun among all the pronouns in a sentence should be useful in pronoun resolution. This is realized in this paper according to its semantic roles (with regard to various predicates in a sentence) and surface position (in a left-to-right order) by mapping each pronoun into 5 levels: a) rank 1 for pronouns with semantic role ARG0/agent of the main predicate; b) rank 2 for pronouns with semantic role ARG1/patient of the main predicate; c) rank 3 for pronouns with semantic role ARG0/agent of other predicates; d) rank 4 for pronouns with semantic role ARG1/patient of other predicates; e) rank 5 for remaining pronouns. Furthermore, for those pronouns with the same ranking level, they are ordered according to their surface positions in a left-to-right order, motivated by previous research on the centering theory (Grosz et al. 1995).
3) Detailed pronominal subcategory features. Given a pronominal expression, its detailed pronominal subcategory features, such as whether it is a first person pronoun, second person pronoun, third person pronoun, neuter pronoun or others, are explored to enhance the discriminative power of both the semantic role features and the relative pronominal ranking feature, considering the predominant importance of pronouns in tracking the local focus structure in discourse.

4.3 Comparison with Previous Work

As a representative in explicitly employing semantic role labeling in coreference resolution, Ponzetto and Strube (2006) explored two semantic role features to capture the predicate-argument structure information to benefit coreference resolution: I_SEMROLE, the predicate-argument pairs of one referring expression, and J_SEMROLE, the predicate-argument pairs of another referring expression. Their experiments on the ACE 2003 corpus shows that, while the two semantic role features much improve the performance of common noun resolution by 3.8 and 2.7 in F-measure on the BNEWS and NWIRE domains respectively, they only slightly improve the performance of pronoun resolution by 0.4 and 0.3 in F-measure on the BNEWS and NWIRE domains respectively.

In comparison, this paper proposes various kinds of centering-motivated semantic role features in attempt to better model the continuity or shift in the local discourse focus by extending the centering theory from the grammatical level to the semantic level. For example, the CAARG0MainVerb feature (as shown in Table 5) is designed to capture the semantic role of the antecedent candidate in the main predicate in modeling the discourse center, while, the ANPronounRanking feature (as shown in Table 5) is designed to determinate the relative priority of the pronominal anaphor in retaining the discourse center.

Although both this paper and Ponzetto and Strube (2006) employs semantic role features, their ways of deriving such features are much different due to different driving forces/motivations behind. As a result, their contributions on coreference resolution are different: while the semantic role features in Ponzette and Strube (2006) captures the predicate-argument structure information and contributes much to common noun resolution and their contribution on pronoun resolution can be ignored, the centering-motivated semantic role features in this paper contribute much in pronoun resolution. This justifies our attempt to better model the continuity or shift of the discourse focus in pronoun resolution by extending the centering theory from the grammatical level to the semantic level and employing the centering-motivated features in pronoun resolution..

5 Experimentation and Discussion

We have evaluated our approach of employing the centering theory in pronoun resolution from the semantic perspective on the ACE 2003 corpus.

5.1 Experimental Setting

The ACE 2003 corpus contains three domains: newswire (NWIRE), newspaper (NPAPER), and broadcast news (BNEWS). For each domain, there exist two data sets, training and devtest, which are used for training and testing respectively. Table 4 lists the pronoun distributions with coreferential relationships in the training data and the test data over pronominal subcategories and sentence distances. Table 4(a) shows that third person pronouns occupy most and neu-

[1] According to the centering theory, the backward-looking center C_b is preferentially realized by a pronoun in the subject position in natural languages, such as English, and people tend to plan ahead and minimize the number of focus shifts.

tral pronouns occupy second while Table 4(b) shows that the antecedents of most pronouns occur within the current sentence and the previous sentence, with a little exception in the test data set of BNEWS.

Pronoun Subcategory	NWIRE		NPAPER		BNEWS	
	Train	Test	Train	Test	Train	Test
First Person	263	103	283	120	455	258
Second Person	61	16	29	36	203	68
Third Person	618	179	919	263	736	158
Neuter	395	151	577	190	482	137
Reflexive	23	6	42	12	26	6
Other	0	0	2	0	2	3

(a) Distribution over pronominal subcategories

Distance	NWIRE		NPAPER		BNEWS	
	Train	Test	Train	Test	Train	Test
0	890	254	1281	347	1149	295
1	447	149	529	197	729	188
2	0	27	0	24	0	41
>2	0	19	0	41	0	**100**
Total	1337	449	1810	609	1878	624

(b) Distribution over sentence distances

Table 4: Pronoun statistics on the ACE 2003 corpus

For preparation, all the documents in the corpus are preprocessed automatically using a pipeline of NLP components, including tokenization and sentence segmentation, named entity recognition, part-of-speech tagging and noun phrase chunking. Among them, named entity recognition, part-of-speech tagging and noun phrase chunking apply the same Hidden Markov Model (HMM)-based engine with error-driven learning capability (Zhou and Su, 2000 & 2002). In particular for SRL, we use a state-of-the-art in-house toolkit, which achieved the precision of 87.07% for ARG0 identification and the precision of 78.97% for ARG1 identification, for easy integration. In addition, we use the SVM-light[2] toolkit with the radial basis kernel and default learning parameters. Finally, we report the performance in terms of recall, precision, and F-measure, where precision measures the percentage of correctly-resolved pronouns (i.e. correctly linked with any referring expression in the coreferential chain), recall measures the coverage of correctly-resolved pronouns, and F-measure gives an overall figure on equal harmony between precision and recall. To see whether an improvement is significant, we also conduct significance testing using paired t-test. In this paper, '>>>', '>>' and '>' denote p-values of an improvement smaller than 0.01, in-between (0.01, 0,05] and bigger than 0.05, which mean significantly better, moderately better and slightly better, respectively.

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

5.2 Experimental Results

Table 5 details various centering-motivated features from the semantic perspective, which are incorporated in our final system. For example, the CAARG0MainVerb feature is designed to capture the semantic role of the antecedent candidate in the main predicate in modeling the discourse center, while the ANPronounRanking feature is designed to determinate the relative priority of the pronominal anaphor in retaining the discourse center. As the baseline, we duplicated the representative system with the same set of 12 basic features, as described in Soon et al (2001). Table 6 shows that our baseline system achieves the state-of-the-art performance of 62.3, 65.3 and 59.0 in F-measure on the NWIRE, NPAPER and BNEWS domains, respectively. It also shows that the centering-motivated features (from the semantic perspective) significantly improve the F-measure by 3.6(>>>), 4.5(>>>) and 7.7(>>>) on the NWIRE, NPAPER and BNEWS domains, respectively. This justifies our attempt to model the continuity or shift of the discourse focus in pronoun resolution via centering-motivated features from the semantic perspective. For comparison, we also evaluate the performance of our final system from the grammatical perspective. This is done by replacing semantic roles with grammatical roles in deriving centering-motivated features. Here, labeling of grammatical roles is achieved using a state-of-the-art toolkit, as described in Buchholz (1999). Table 6 shows that properly employing the centering theory in pronoun resolution from the grammatical perspective can also improve the performance. However, the performance improvement of employing the centering theory from the grammatical perspective is much lower, compared with that from the semantic perspective. This validates our attempt of employing the centering theory in pronoun resolution from the semantic perspective instead of from the grammatical perspective. This also suggests the great potential of applying the centering theory in pronoun resolution since the centering theory is a local coherence theory, which tells how subsequent utterances in a text link together.

Table 7 shows the contribution of the semantic role features and the relative pronominal ranking feature in pronoun resolution when the detailed pronominal subcategory features are included:

Feature category	Feature	Remarks
Semantic Role-based Features	CAARG0	1 if the semantic role of the antecedent candidate is ARG0/agent; else 0
	CAARG0MainVerb	1 if the antecedent candidate has the semantic role of ARG0/agent for the main predicate of the sentence; else 0
	ANCASameTarget	1 if the anaphor and the antecedent candidate share the same predicate with regard to their semantic roles; else 0
Relative Pronominal Ranking Feature	ANPronounRanking	Whether the pronominal anaphor is ranked highest among all the pronouns in the sentence
Detailed Pronominal Subcategory Features	ANPronounType	Whether the anaphor is a first person, second person, third person, neuter pronoun or others
	CAPronounType	Whether the antecedent candidate is a first person, second person, third person, neuter pronoun or others

Table 5: Centering-motivated features incorporated in our final system
(with AN indicating the anaphor and CA indicating the antecedent candidate)

System Variation	NWIRE			NPAPER			BNEWS		
	R%	P%	F	R%	P%	F	R%	P%	F
Baseline System	*57.0*	***68.6***	*62.3*	*61.1*	*70.1*	*65.3*	*49.0*	*73.9*	*59.0*
Final System (from the semantic perspective)	***64.1***	*67.8*	***65.9***	***67.5***	***72.4***	***69.8***	***59.9***	***75.3***	***66.7***
Final System (from the grammatical perspective, for comparison)	*63.3*	*64*	*63.6*	*64.7*	*68.8*	*66.7*	*57.1*	*70.1*	*63.1*

Table 6: Contributions of centering-motivated features in pronoun resolution

System Variation	NWIRE			NPAPER			BNEWS		
	R%	P%	F	R%	P%	F	R%	P%	F
Baseline System	57.0	**68.6**	62.3	61.1	70.1	65.3	49.0	73.9	59.0
+SR and DC	**64.8**	67.8	**66.3**	67.2	**72.9**	**69.9**	59.1	**75.3**	66.3
+PR and DC	61.5	65.4	63.4	64.9	72.1	68.3	57.4	73.5	64.5
+SR, PR and DC (Final System)	64.1	67.8	65.9	**67.5**	72.4	69.8	**59.9**	**75.3**	**66.7**

Table 7: Contribution of the semantic role features (SR) and the relative pronominal ranking feature (PR) in pronoun resolution when the detailed pronominal subcategory features are included

1) The inclusion of the semantic role features improve the performance by 4.0(>>>), 4.6(>>>) and 7.3(>>>) in F-measure on the NWIRE, NPAPER and BNEWS domains, respectively. This suggests the impact of semantic role information in determining the local discourse focus. Since pronouns preferentially occur in the subject position and tend to refer to the main subject (Ehrlich 1980; Brennan 1995; Walker et al. 1998; Cahn 1995; Gordon and Searce 1995; Kibble et al. 2001), this paper only applies semantic features related with the semantic role of ARG0/agent, which is closely related with the grammatical role of subject, with regard to various predicates in a sentence. We have also explored features related with other semantic roles. However, our preliminary experimentation shows that they do not improve the performance, even for ARG1/patient, and thus are not included in the final system. This may be due to that other semantic roles are not discriminative enough to make a difference in deciding the local discourse structure.
2) It is surprising to notice that further inclusion of the relative pronominal ranking feature has only slight impact (slight positive impact on the BNEWS domain and slight negative impact on the NWIRE and NPAPER domains) on the ACE 2003 corpus. This suggests that most of information in the relative pronominal ranking feature has been covered by the semantic role features. This is not surprising since the semantic role of ARG0/agent, which is explored to derive the semantic role features, is also applied to decide the relative pronominal ranking feature.

The inclusion of the relative pronominal ranking feature improve the performance by 1.1(>>>), 3.0(>>>) and 5.5(>>>) in F-measure. Our further evaluation reveals that the performance improvement difference among different domains of the ACE 2003 corpus is due to the distribution of pronouns' antecedents occurring over different sentence distances, as shown in

Table 4. This suggests the usefulness of the relative pronominal ranking feature in resolving pronominal anaphors over longer distance. This is consistent with our observation that, as the percentage of pronominal anaphors referring to more distant antecedents increase, its impact turns gradually from negative to positive, when further including the relative pronominal ranking feature after the semantic role features. The reason that we include the detailed pronominal subcategory information is due to predominant importance of pronouns in tracking the local focus structure in discourse and that such detailed pronominal subcategory information is discriminative in tracking different subcategories of pronouns. This suggests the usefulness of considering the distribution of the local discourse focus over detailed pronominal subcategories. One interesting finding in our preliminary experimentation is that the inclusion of the detailed pronominal subcategory features alone even harms the performance. This may be due to the reason that the detailed pronominal subcategory features do not have the discriminative power themselves and that the semantic role features and the relative pronominal ranking feature provide an effective mechanism to explore the role of such detailed pronominal subcategory features in helping determine the local discourse focus.

	Pronoun Subcategory	NWIRE	NPAPER	BNEWS
Baseline System	First Person	55.7	55.9	56.6
	Second Person	54.6	60.4	44.0
	Third Person	72.6	80.9	75.7
	Neuter	41.5	50.4	50.2
	Reflexive	**85.7**	**70.0**	60.0
	Total	*62.3*	*65.3*	*59.0*
Final System	First Person	**64.7**	**67.0**	**65.6**
	Second Person	**78.6**	**70.0**	**51.9**
	Third Person	**80.9**	**81.8**	**80.4**
	Neuter	**48.3**	**53.0**	**58.3**
	Reflexive	71.4	66.7	**80.0**
	Total	***65.9***	***69.8***	***66.7***

Table 8: Performance comparison of pronoun resolution in F-measure over pronoun subcategories

Table 8 shows the contribution of the centering-motivated features over different pronoun subcategories. It shows that the centering-motivated features contribute much to the resolution of the four major pronoun subcategories (i.e. first person, second person, third person and neuter) while its negative impact on the minor pronoun subcategories (e.g. reflexive) can be ignored due to their much less frequent occurrence in the corpus. In particular, the centering-motivated features improve the performance on the major three pronoun subcategories of third person / neuter / first person, by 8.3(>>>)/6.8(>>>)/9.0(>>>), 0.9(>>)/ 2.6 (>>>)/11.1(>>>) and 4.7(>>>)/8.1(>>>)/9.0 (>>>), on the NWIRE, NPAPER and BNEWS domains of the ACE 2003 corpus, respectively.

	Distance	NWIRE	NPAPER	BNEWS
Baseline System	<=0	61.6	64.5	68.7
	<=1	60.4	67.5	60.0
	<=2	62.9	67.4	63.7
	Total	*62.3*	*65.3*	*59.0*
Final System	<=0	**64.3**	**70.3**	**78.7**
	<=1	**66.8**	**72.3**	**72.5**
	<=2	**66.6**	**71.8**	**71.8**
	Total	***65.9***	***69.8***	***66.7***

Table 9: Performance comparison of pronoun resolution in F-measure over sentence distances

Table 9 shows the contribution of the centering-motivated features over different sentence distances. It shows that the centering-motivated features improve the performance of pronoun resolution on different sentence distances of 0/1/2, by 2.7(>>>) / 5.8(>>>) / 10.0 (>>>), 6.4(>>>) / 4.8(>>>) / 12.5(>>>) and 3.7 (>>>)/4.4(>>>)/8.1(>>>), on the NWIRE, NPAPER and BNEWS domains of the ACE 2003 corpus, respectively. This suggests that the centering-motivated features are helpful for both intra-sentential and inter-sentential pronoun resolution.

6 Conclusion and Further Work

This paper extends the centering theory from the grammatical level to the semantic level and much improves the performance of pronoun resolution via centering-motivated features from the semantic perspective. This is mainly realized by employing various semantic role features with regard to various predicates in a sentence, in attempt to model the continuity or shift of the local discourse focus. Moreover, the relative ranking feature of a pronoun among all the pronouns is explored to help determine the relative priority of the pronominal anaphor in retaining the local discourse focus. Evaluation on the ACE 2003 corpus shows that both the centering-motivated semantic role features and pronominal ranking feature much improve the performance of pronoun resolution, especially when the detailed pronominal subcategory features of both the anaphor and the antecedent candidate are included. It is not surprising due to the predomi-

nance of pronouns in tracking the local discourse structure in a text.

To our best knowledge, this paper is the first research which successfully applies the centering-motivated features in pronoun resolution from the semantic perspective.

For future work, we will explore more kinds of semantic information and structured syntactic information in pronoun resolution. In particular, we will further employ the centering theory in pronoun resolution from both grammatical and semantic perspectives on more corpora.

Acknowledgement

This research is supported by Project 60673041 under the National Natural Science Foundation of China, project 2006AA01Z147 under the "863" National High-Tech Research and Development of China, project 200802850006 under the National Research Foundation for the Doctoral Program of Higher Education of China, project 08KJD520010 under the Natural Science Foundation of the Jiangsu Higher Education Institutions of China.

References

C. Aone and W.W. Bennett. 1995. Evaluating automated and manual acquisition of anaphora resolution strategies. *ACL*'1995:122-129.

S.E. Brennan, M.W. Friedman, and C.J. Pollard. 1987. A centering approach to pronoun. *ACL'1987:* 290-297.

S.E. Brennan 1995. Centering attention in discourse. *Language and Cognitive Process*, 10/2: 137-67.

S. Buchholz, J. Veenstra and W. Daelemans. 1999. Cascaded Grammatical Relation Assignment. *EMNLP-VLC'1999*: 239-246

S. Cote. 1983. Ranking and forward-looking centers. *In Proceedings of the Workshop on the Centering Theory in Naturally-Occurring Discourse*. 1993.

D.M. Carter. 1987. Interpreting Anaphors in Natural Language Texts. *Ellis Horwood*, Chichester, UK.

J. Carbonell and R. Brown. 1988. Anaphora resolution: a multi-strategy approach. *COLING'1988*: 96-101.

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

P.C. Gordon, B.J. Grosz and L.A. Gilliom. 1993. Pronouns, names and the centering of attention in discourse. *Cognitive Science*.1993.17(3):311-348

P.C. Gordon and K. A. Searce. 1995. Pronominalization and discourse coherence, discourse structure and pronoun interpretation. *Memory and Cognition*. 1995.

S. Harabagiu and S. Maiorano. 2000. Multiligual coreference resolution. *ANLP-NAACL* 2000:142-149.

A.K. Joshi and S. Weinstein. 1981. Control of inference: Role of some aspects of discourse structure-centering. *IJCAI'1981*:385-387

R. Kibble. 2001. A Reformulation of Rule 2 of Centering. *Computational Linguistics*, 2001,27(4): 579-587

M. Kameyama. 1986. Aproperty-sharing constraint in centering. *ACL* 1986:200-206

M. Kameyama. 1988. Japanese zero pronominal binding, where syntax and discourse meet. *In Proceeding of the Second International Workshop on Japanese Syntax.* 1988.

R. Mitkov. 1998. Robust pronoun resolution with limited knowledge. *COLING-ACL*'1998:869-875.

A. Moschitti and S. Quarteroni. 2008. Kernels on linguistics structures for answer entraction. *ACL*'08:113-116

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. *ACL'2002*: 104-111

V. Ng. 2007. Semantic Class Induction and Coreference Resolution. *ACL*'2007 536-543.

M. Palmer, D. Gildea and P. Kingsbury. 2005. The proposition bank: A corpus annotated with semantic roles. *Computational Linguistics,* 31(1):71-106.

S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J.H. Martin, and D. Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning,* 2005.60:11-39.

S. P. Ponzetto and M. Strube. 2006. Semantic Role Labeling for Coreference Resolution. *EMNLP*'2006 143-146.

E. F. Prince and M. A. Walker. 1995. A bilateral apporach to givenness: a hearer-status algorithm and a centering algorithm. *In Proceedings of 4th International Pragmatics Conference*.

E. Rich and S. LuperFoy. 1988. An architecture for anaphora resolution. *In Proceedings of the 2nd Conference on Applied Natural Language Processing. ANLP*'1988: 18-24.

W.M. Soon, H.T. Ng and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrase. *Computational Linguistics,* 2001, 27(4):521-544.

D. Shen and M. Lapata. 2007. Using semantic roles to improve question answering. *EMNLP-CoNIL* 2007:12-21

C. Sidner. 1979. Toward a computation of intrasentential coreference. *Technical Report TR-537,MIT. Artificial Intelligence Laboratory*.

C. Sidner. 1981. Focusing for interpretation of pronouns. *Computational Linguistics*,1981.7:217-231

J. Tetreault. 1999. Analysis of syntax-based pronoun resolution methods. *ACL* 1999:602-605

J. Tetreault. 2001. A corpus-based evaluation of centering and pronoun resolution. *Computational Linguistics*. 2001. 27(4):507-520.

M. Walker, A. K. Joshi and E. Prince. 1998. Centering in naturally occurring discourse: *An overview. Clarendon Press*:1-28

X.F. Yang, J. Su, G.D. Zhou and C.L. Tan. 2004. Improving pronoun resolution by incorporating coreferential information of candidates. *ACL'*2004:127-134.

X.F. Yang, J. Su and C.L. Tan. 2005. Improving Pronoun Resolution Using Statistics - Based Semantic Compatibility Information. *ACL'*2005:165 -172.

X.F. Yang and J. Su. 2007. Coreference Resolution Using Semantic Relatedness Information from Automatically Discovered Patterns. *ACL'2007*: 528-535.

G.D. Zhou and J. Su. 2004. A high- performance coreference resolution system using a multi- agent strategy. *COLING'* 2004:522- 528.

Y. Ziv and B.J. Grosz. 1994. Right dislocation and attentional state. *Israel Association of Theoretical Linguistics Meetings*'1994. 184-199.

Person Cross Document Coreference with Name Perplexity Estimates

Octavian Popescu
popescu@racai.ro

.

Abstract

The Person Cross Document Coreference systems depend on the context for making decisions on the possible coreferences between person name mentions. The amount of context required is a parameter that varies from corpora to corpora, which makes it difficult for usual disambiguation methods. In this paper we show that the amount of context required can be dynamically controlled on the basis of the prior probabilities of coreference and we present a new statistical model for the computation of these probabilities. The experiment we carried on a news corpus proves that the prior probabilities of coreference are an important factor for maintaining a good balance between precision and recall for cross document coreference systems.

1 Introduction

The Person Cross Document Coreference (Grishman 1994) task, which requires that all and only the textual mentions of an entity of type Person be individuated in a large collection of text documents, is one of the challenging tasks for natural language processing systems. In the most general case the corpus itself is the only available source of information regarding the persons mentioned and we consider that this is the case in this paper. A PCDC system must be able to use the information existing in the corpus in order to assign to each personal name mention (PNM) a piece of context. The coreference of any two PNMs is decided mainly on the basis of the similarity of the pieces of contexts associated with them. A successful PCDC must accurately extract the relevant context for coreference.

However, the context relevance is not absolute. Whether the contextual information uniquely individuates a person is a matter of probability. This paper presents a statistical technique developed to provide a PCDC system with more information regarding the probability of a correct coreference. The reason for developing this technique is twofold: (i) the relevant coreference context depends on the corpus itself and (ii) valid coreferences require a large amount of information, which is unavailable in the majority of cases.

The first reason is linked to a particularity of the CDC task that makes it more complex than other NLP tasks. Unlike in other disambiguation tasks, in the CDC tasks the relevant coreference context depends on the corpus itself. In word sense disambiguation, for instance, the distribution of the relevant context is mainly regulated by strong syntactic and semantic rules. The existence of such rules makes it possible for the disambiguation decisions to be made considering the local context. On the other hand, the distribution of the PNMs in a corpus is rather random and the relevant coreference context is a dynamic variable depending on the diversity of the corpus, that is, on how many different persons with the same name share a similar context. To exemplify, consider the name "John Smith" and an organization, say "U.N.". The extent to which "works for U.N." in "John Smith works for U.N." is a relevant coreference context depends on the diversity of the corpus itself. If in that corpus, among all the "John Smiths" there is only one person who works for "U.N." then "works for U.N." is a relevant coreference context, but if there are many "John Smiths" working for U.N., then "works for U.N." is not a relevant coreference system; in this last case, more contextual evidence is needed in order to correctly corefer the "John Smith" PNMs. The relevance of a context for coreference also depends on the corpus, not only on the specific relationship that exists between "John Smith" and

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 997–1006,
Singapore, 6-7 August 2009.

"works for U.N.". Thus, A PCDC system must have access to global information regarding the PNMs.

The second reason comes from practical considerations. The amount of information required to correctly infer PNMs coreferences is not present in corpus in a computationally friendly way. In many cases the relevant coreference information is embedded in semantic and ontological deep inferences, which are difficult to program In as much as 60% of the cases, two documents containing the same name, from a news corpus, lack contexts which are directly similar and big enough to correctly decide on the coreference.

We propose a new method to control the amount of contextual coreference required for correct coreferences. Rather than having fixed rules deciding on the size of the context surrounding a PNM, we propose a probabilistic approach that requires contextual evidence for coreference differentially, by considering the prior probability of the coreference of two PNMs; the higher this probability is, the less their correct coreference depends on the context and vice versa. We present a statistical model where the prior coreference probabilities are computed considering only the corpus itself, and we show how these probabilities are used by a PCDC system that dynamically revises the amount of context relevant for coreference.

In Section 2 we review the CDC relevant literature. In section 3 we analyze the data from annotated coreference corpora and we individuate a specific problem, setting up a working hypothesis. In Section 4 we develop a statistical model for computing the prior coreference probabilities and in Section 5 we present the results obtained by applying it to a large news corpus. In section 6 a direct evaluation on CDC is carried on a test corpus. In Section 7 we show how the proposed techniques extends naturally to a strategy of construction relevant test corpora for CDC task. The paper ends with the Conclusion and the Future Research section.

2 Related Work

In a classical paper (Bagga and Baldwin 1998), a PCDC system based on the vector space model (VSM) is proposed. While there are many advantages in representing the context as vectors on which a similarity function is applied, it has been shown that there are inherent limitations associated with the vectorial model (Popescu 2008). These problems, related to the density in the vectorial space (superposition) and to the discriminative power of the similarity power (masking), become visible as more cases are considered.

Testing the system on many names, (Gooi and Allan, 2004), it has been noted empirically that the accuracy of the results varies significantly from name to name. Indeed, considering just the sentence level context, which is a strong requirement for establishing coreference, a PCDC system obtains a good score for "John Smith". This happens because the prior probability of coreference of any two "John Smiths" mentions is low, as this is a very common name and none of the "John Smith" has an overwhelming number of mentions. But for other types of names the same system is not accurate. If it considers, for instance, "Barack Obama", the same system obtains a very low recall, as the probability of any two "Barack Obama" mentions to corefer is very high and the relevant coreference context is found very often beyond the sentence level. Without further adjustments, a vectorial model cannot resolve the problem of considering too much or too little contextual evidence in order to obtain a good precision for "John Smith" and simultaneously a good recall for "Barack Obama".

In an experiment using bigrams (Pederson et al. 2005) on a news corpus, it has been observed that the relationship between the amount of information given to a PCDC system and the performances is not linear. If the system has received in input the correct number of persons with the same name, the accuracy of the system has dropped. A typical case for this situation is when there is a person that is very often mentioned, and few other persons having few mentions; when the number of clusters is passed in the input, the clusters representing the persons who are rarely mentioned are wrongly enriched. However, this situation can be avoided if there is a measure of how probable it is to have a certain number of different persons with the same name, each being mentioned very often in a newspaper.

Recently, there has been a major interest in the PCDC systems, and, in the last two years, three important evaluation campaigns have been organized: Web People Search-1 (Artiles et al. 2007), ACE 2008 (www.nist.gov/speech/tests/ace/). It has been noted that the data variance between training and test is very high (Lefever 2007). Rather than being a particularity of those corpora, the problem is general. The performances of a bag of words VSM depends to a very high extent on the corpus diversity (see Section 3).

For reliable results, a PCDC system must have access to global information regarding the coreference space.

Rich biographic facts have been shown to improve the accuracy of PCDC (Mann and Yarowsky 2003). Indeed, when available, the birth date, the occupation etc. represent a relevant coreference context because the probability that two different persons have the same name, the same birth date and the same occupation is negligible. However, it is equally unlikely to find this information in a news corpus a sufficient number of times. Even for a web corpus, where the amount of this kind of information is higher than in a news corpus, the extended biographic facts, including e-mail address, phones, etc., contribute only with approximately 3% to the total number of coreferences (Elmacioglu et al. 2007).

In order to improve the performances of the PCDC systems based on VSM, some authors have focused on methods that allow a better analysis of the context by extracting the dependency chains (Ng 2007). The special importance of pieces of context has been exploited by implementing a cascade clustering technique (Wei 2006). Other authors have relied on advanced clustering techniques (among others Han et al. 2005, Chen 2006). However, these techniques rely on the precise analysis of the context, which is a time consuming process. It has been also noted that, in spite of deep analysis, the relevant coreference context is hard to find (Vu 2007).

The technique we present in the next sections is complementary to these approaches. We propose a statistical model designed to offer to the PCDC systems information regarding the distribution of PNMs in the corpus. This information is used to reduce the contextual data variation and to attain a good balance between precision and recall.

3 Data Analysis

In this Section we present the data analysis of the PNMs. We are interested in establishing a relationship between the distribution of the PNMs and the relevant context for coreference. As mentioned in the preceding sections, the amount of the relevant context for coreference cannot be decided prior to the investigation of that particular corpus. The performances of a bag of words VSM with a prior defined context approach will vary greatly from corpus to corpus. We have run the following experiment: we have considered the training and test corpora used in Web People Search-1 (WePS-1), which are web page corpora, and we have implemented a bag of word approach with two variants of clustering: agglomerative (A), and hierarchic (H). We have randomly chosen a set of seven names from training and test (14 names in total) and we have compared the results applying the two systems, A and H, on each set of names. In Figure 1 we present the results obtained. The figures on the vertical axes are computed using $F_{\alpha=0.5}$ formula.

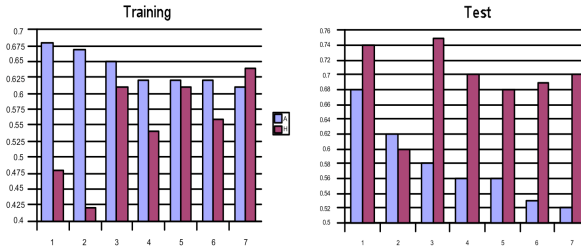


Figure 1. Variation between training and test

We have noticed a great variation in the behavior of the two systems. In order to search for an explanation for this difference we have looked at the distribution in the two corpora of the Named Entities, of the words denoting professions and of the meta-contextual information - e-mails, urls, phones, and addresses. It turns out that these types of contextual information are distributed between training and test approximately evenly. (see Table 1a,b).

Profession	training occ.	test occ.
Doctor	543	668
Lawyer	277	385
Professor	523	490
Researcher	340	166
Teacher	617	569
Coach	467	471
Actor	998	790

Table 1a. Profession words in training and test

Address	training occ.	test occ.
Phone	1,109	1,169
Fax	606	426
e-mail	3,134	2,186

Table 1b. Meta-Context in training and test

By manually investigating the training and test set of our experiment we have reached the conclusion that the reason for the difference is two fold: firstly, while the distribution of the words denoting profession is similar, in the test set the modifiers, for example “internist”, “neurosurgeon” for “doctor”, are more frequent. Secondly, the number of different persons having the same

name is, on average, higher in test than in training. The results plotted in Figure 1 show that it is not a question of which algorithm is better, but rather that there are different cases where one approach is preferred over the other. The problem we face is deciding when it is appropriate to use one or the other.

To induce from the corpus itself when a piece of context is or is not a relevant context requires deep ontological inferences and a very powerful tool of semantic analysis of the context. Consider for example two words denoting profession, "doctor" and "researcher", and their possible modifiers "internist", "neurosurgeon", and "professor" and "PhD". In the first case it is certain that the coreference is not possible, while in the second the coreference is very probable. To find out such relationships is computationally very hard. However, the analysis carried out further shows that we can avoid making such computations in most of the cases.

The number of different persons is a parameter that cannot be known beforehand. However, not all the names behave alike with respect to coreference. There are noticeable differences between names; for example less than 5 000 first names cover approximately 96% of the total of first names, while for the same percentage of coverage more than 70 000 of last names must be considered (Popescu et al. 2007). Let us call perplexity of a name the number of different persons that carry it. The search space depends directly on the name perplexity. The bigger the perplexity, the larger the amount of information required for the correct coreference must be. It seems natural that the amount of contextual evidence required by a PCDC depends on the name perplexity.

In order to evaluate the relationships between the context and the name perplexity, we need an annotated corpus. We have used the I-CAB corpus (Magnini et al. 2006), which is a four-day news corpus fully annotated, coreference relationships included. The documents in this corpus are entire pieces of news. For each PNM we have counted how many contexts containing specific information about the person carrying the respective name is present in that particular document. There are many types of contexts that refer to a person, but some of these types are very infrequent. We considered only those types of information that are present at least 5% of the times in the context surrounding a PNM. Table 2 presents the results of this investigation.

	occ.	diff occ	entities
First Names	2299	676	1592
Last Names	4173	1906	2191
Middle Name	110	44	41
Activity	973	322	569
Affiliation	566	399	409
Role	531	211	317
Family Relation	133	46	94

Table 2. Name perplexity and context

On the second column the total number of occurrences is listed, on the third column how many of these occurrences have different values (no case sensitive string match), and on the fourth column the number of different persons (Entities) having that information. The entries "activity", "affiliation", and "role" represent pieces of context where the respective information is directly expressed (no inferences). We call this type of context professional context and for approximately 30% of the PNMs, one of the above three types of professional contexts is present.

The perplexity of the first names, computed as the ratio between the fourth column and the third column is two times bigger than the perplexity of the last names. The lowest name perplexity is obtained by the names having a middle name - a name with at least three tokens – and it is very close to 1 (1.07). Comparatively, the highest perplexity of two tokens name is 3. The relationship between the number of tokens of a name and its perplexity is straightforward: for names with more than four tokens the perplexity is 1 in 99,6% of the cases (the name by itself is a relevant context for coreference).

In approximately 74% of the cases there is just one entity corresponding to a two-token name. Considering any two PNMs of the same name the similarity of two of the professional contexts guarantees the correct coreference. However, two professional contexts are present in only 4% of the cases. There are just four cases when considering just one professional attribute was misleading, and all these cases are high perplexity names. Moreover, in the case of many low perplexity names, the contexts could be minimally similar in order to correctly corefer any two PNMs of that respective name.

This analysis shows that there is a direct relationship between the name perplexity and the relevant coreference context. However, the average figures are not very informative, as the variance of perplexity is very high. Rather than fo-

cusing on the exact figure for name perplexity, we will try to partition the names according to their perplexity and to link each partition to a specific behavior with respect to coreference. The partitioning technique should ensure that the variance of the name perplexity within the same partition is low and that a specific amount of context should lead to the correct coreference decision for the great majority of names within that partition.

Our working hypothesis is that we can estimate the name perplexity within each partition and use this information to control the amount of contextual evidence required. Let us recall the "John Smith" and "Barack Obama" example from the previous section. Both "John" and "Smith" are American common first and last names. The chance that many different persons carry this name is high. On the other hand, as both "Barack" and "Obama" are rare American first and last names respectively, almost surely many mentions of this name refer only to one person. The argument above does not depend on the context, but just on the prior estimation of the usage of those names. Having an estimation of a name's perplexity, we may decrease/increase the amount of contextual evidence needed.

4 (p, γ) Statistical Model

Let $\mathcal{D}$ be the set of all PNMs from a given corpus $\mathcal{C}$ and let $\mathcal{D}_N$ be the set of corresponding names. We want to find a partition $\mathcal{P}$ of $\mathcal{D}_N$ such that within each partition the name perplexity varies only within predicted margins. Let X be a random variable with uniform distribution over $\mathcal{D}_N$ and let Y be the random variable defined by X's name perplexity. Let us suppose that we want $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$ to be a partition of $\mathcal{D}_N$, where the percentage of each partition class is p_i: the first partition class contains p_1 percentage of the name population, the second partition class contains p_2 percentage of the name population and the last partition class contains $p_m = 1 - \Sigma p_i$ percentage of the name population.

If we knew the distribution function of Y, let's call it F, we would simply determine ξ_i from equation 1, where $P_i = \Sigma p_k$, $k \le i$:

$$\xi_i = F^{-1}(P_i) \Leftrightarrow F(Y<\xi_i) = P_i \qquad (1)$$

and we would know that in each partition p_i the name perplexity is between ξ_{i-1} and ξ_i, with $\xi_0 = 0$. However we do not know F. Fortunately, we can estimate ξ_i.

There is no restriction that may impose a particular form for F; for example, the normal distribution hypothesis of name perplexity is ruled out by a χ^2 test with 96.5% confidence for the 14 names chosen from WePS-1 (see Section 3, first paragraph).

We are going to present a distributional free method for constructing the partition $\mathcal{P}$. The advantage of this method is that it does not depend on any assumption about the PNMs distribution.

Let us consider $X_1, X_2, \ldots, X_n$ a sample of independent and identical distributed names from $\mathcal{D}_N$. By rearranging the indexes, without losing the generality, let us suppose that $Y_1, Y_2, \ldots, Y_n$ is ordered, that is $Y_1 \le Y_2 \le \ldots \le Y_n$. Even if we do not know what form F has, we can still use equation (1) in order to estimate ξ_i. The expected value of $F(Y_i)$ is (Hogg, Mckean, Craig 2006):

$$E[F(Y_i)] = i/(n+1) \qquad (2)$$

which is an estimation of how much mass probability is on the left of Y_i. In our terms, we estimate that $E[F(Y_i)]$ percentage of the name population has a name perplexity lower than Y_i.

For a given number ξ, the percentage of the name population having the name perplexity at most ξ is determined by finding the smallest Y_i greater than ξ and use the equation (2) to estimate $E[F(Y_i)]$.

In order to build the partition $\mathcal{P}$ we are interested in the percentage of the name population that has the perplexity between two given values. Let (Y_i, Y_j) be the smallest interval that includes these two values. We can estimate the percentage of the name population that has a perplexity between Y_i and Y_j. This estimate is simply $F(Y_j) - F(Y_i)$. We can use directly equation (2) to estimate this difference. However, it is more important to have a confidence interval for this estimate, that is we want to know what the probability is that the interval (F(Yi), F(Yj)) contains at least a given percentage of the population, p. The optimal partition $\mathcal{P}$ is the one that maximizes the confidence in the fact that within each of the partition classes as many names as possible have the name perplexity in a given interval.

Let p be a given real number between (0,1) representing the mass probability that goes into the interval $(F(Y_i), F(Y_j))$. Let $\gamma = P(F(Y_j) - F(Y_i) \ge p)$. Fortunately γ has a distribution that does not depend on F. More precisely, γ has a beta distribution given by the function in formula (3):

$$\gamma = P(F(Y_j) - F(Y_i) \geq p) =$$

$$\int_p^1 \Gamma(n+1)/(\Gamma(j-i))\,\Gamma(n-j+i+1)x^{j-i-1}(1-x)^{n-j+i}dx \quad (3)$$

The Γ, called the gamma function, is the extension of the factorial, $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$. The gamma function has the property that $\Gamma(x) = \Gamma(x-1)\,\Gamma(x-2)\ldots\,\Gamma(1)$; for x an integer, as the arguments in the formula (3) are, $\Gamma(x) = (x-1)!$

The formula (3) gives us a method of building the partition $\mathcal{P}$. Let us start with a set of perplexity intervals: $(\xi_0, \xi_1]$, $(\xi_1,\xi_2]$, $\ldots(\xi_{m-1}, \xi_m]$. We partition the names in $\mathcal{D}_N$ such that we maximize the confidence γ that at least p_i percentage of the name population has a name perplexity in $(\xi_{i-1}, \xi_i]$. We chose an independent and identical distributed sample X of names to which the ordered sample Y of name perplexity values corresponds. We start with the lowest perplexity interval and determine p_1,γ_1 and Y_0, Y_{i1}, such that $Y_0 \leq \xi_0 \leq \xi_1 \leq Y_{i1}$ and $\gamma_1 = P(F(Y_{i1}) - F(Y_0) \geq p_1)$. The ith index varies according to the desired γ_1, when p_1 is given, and vice-versa. We can choose i1 with m-1 liberty grades. Once we are satisfied with the values (p_1,γ_1), we search for the i2th index such that $Y_{i1} \leq \xi_1 \leq \xi_2 \leq Y_{i2}$ and (p_{i2},γ_{i2}) have the desired value. The process continues till the penultimate (p_{m-1},γ_{m-1}). We have no liberty in choosing the (p_m,γ_m).

We can compute the size of the sample needed for guaranteeing a minimum γ and p.

Let us give an example. Suppose that $(\xi_0, \xi_1] = (0,2]$. Thus we are interested in finding p, the percentage of the name population such that we can be γ sure that at least p names have a perplexity between 1 and 2 inclusive. We take a random sample of n = 30 and suppose the smallest index i1 such that $Y_i \geq 3$ for all $i > i1$ is 17 . We want to compute the confidence γ that at least p = 60% of the name population has the name perplexity within (0,2]:

$$\gamma = 1 - \int_0^{0.6} 30!/(16!15!)x^{15}(1-x^{14})dx =$$

$$= 1 - k(\int_0^{0.6} x^{15}dx + \int_0^{0.6} x^{29}dx) =$$

$$= 1 - k[(1/16)(6/10)^{16} + (1/30)(6/10)^{30}]$$

$$\geq .965$$

In practice we want to have optimal values for p and γ; a large p implies a small γ and vice-versa. The optimality is determined by the accuracy of the CDC system: we want to have the largest possible percentage of names into each partition such that our confidence that the names inside each partition have the same perplexity.

It is useful to work the equation (1) backwards. Suppose that we established the first partition class of $\mathcal{P}$ - we have found the i1th index, p_1, and γ_1. Now we refer only to the names in the partition class. We can compute the probability that a certain percentage of the names *within that particular partition class* have a given name perplexity. That is, we consider a random *sample inside the partition class,* X, and its correspondent random variable Y, as above. The confidence that $p_{1inside}$ percentage of names have the name perplexity ξ_p within the interval (Y_0, Y_{ith}) is:

$$P(Y_0 < \xi_p < Y_{i1}) = \Sigma_k \binom{n}{k} p^k(1-p)^{n-k} \quad (4)$$

$\binom{n}{k}$ represents the k-combinations of size n.

By taking advantage of the bootstrapping method (Efron and Tibshirani 1993) we do not have to resample inside the partition class. We use the $Y_0, .., Y_{i1}$ values with replacement. Using (4) we obtain $p_{1inside}$ which shows us which percentage inside the partition class has the name perplexity within $(Y_0, Y_{i1}]$. And consequently we can compute $\gamma_{1inside}$. Finally we are able to formulate the following statement about each partition class:

> In the ith partition class enter p_i percentage of the name population with a confidence γ_i. Inside this partition class we are $\gamma_{iinside}$ confident that $p_{iinside}$ percentage of the names have a name perplexity within $(Y_{i1-1}, Y_{i1}]$.

The p and γ indicate the theoretical values that define the partition. In practice the exact distribution of the names into the subset is unknown, therefore each heuristics that computes the perplexity creates its own distribution. The values $\gamma_{iinside}$ and $p_{iinside}$ control how much a certain heuristics departs from the theoretical values. The optimal heuristics have very big figures for $\gamma_{iinside}$ and $p_{iinside}$.

In the next section we present an experiment carried on a news corpus. We show how the above model leads to a stable partition of names and that inside each partition class reliable (p,γ) values can be computed.

5 Name Perplexity Partition

For the experiment described in this section, we have used a two-year part of the seven-years Italian local newspaper corpus called Adige500k Corpus (Magnini 2006).

We describe below how we compute the perplexity class for the one-token names and two-tokens names respectively. As mentioned in section 3, the name perplexity decreases rapidly for tree-token or more names. If desired, the same technique could also be applied for those names. In Adige500k there are 106, 187 different one-token names; 429, 243 two-token names; 36, 773 three-token names; 5, 152 four-token names, 940 four token names and less than 300 different four-token or more names.

An estimate of the name perplexity of the one-token names is the size of the different one-token names with which it forms a complete PNM in the corpus. For example for the first name "John" the estimation of its perplexity is the size of the one-token last names it combines with in forming PNMs, like "Smith, Travolta, Kennedy" etc. The bigger the size of its complementary names, the higher is its name perplexity. In Table 3 we present the figures of these estimates.

occurrences (interval)	average perplexity
1-5	4.13
6-20	8.34
21-100	17.44
101-1,000	68.54
1,000-5,000	683.95
5,000-31,091	478.23

Table 3. Average perplexity one-token names

We start with a five name perplexity classes: "very low" (VL) , "low" (L) , "medium", (M) "high" (H) and "very high" (VH). The name perplexity of a two-token name is interpolated from the name perplexity of its components. We used the following heuristics: the name perplexity class is the average name perplexity classes of its one-token name. If the name perplexity classes are the same then the name perplexity class of the whole name is one class less (if possible).

In order to compute the borderline between two consecutive classes we apply the (p, γ) method. We selected 25 two-tokens names and we manually investigate their occurrence in order to know their real name perplexity. The perplexity classes obtained after applying the (p, γ) technique are listed in Tables 4a and 4b respectively.

perplexity class	percentage
very high (VH)	5.3%
high (H)	8.7%
medium (M)	20.9%
low (L)	27.6%
very low (VL)	37.5%

Table 4a. First Name perplexity classes

perplexity class	percentage
very high (VH)	1.8%
high (H)	3.36%
medium (M)	17.51%
low (L)	20.31%
very low (VL)	57.02%

Table 4b. Last Name perplexity classes

Tables 4a and 4b fully describe the partition for one-token names. Ordering the one token names according to their perplexity we chose the first ones according to the percentage listed above. The same process applies to the one-token last names. The values computed for two-token names are listed.

	P	γ	p_{inside}	γ_{inside}
VH	0.04%	70%	70%	80%
H	2.53%	76%	70%	80%
M	10.08%	87%	80%	82%
L	27.97%	90%	99%	90%
VL	59.38%	96.5%	99%	96.5%

Table 5. (p, γ, p_{inside}, γ_{inside}) values

6 CDC with Name Perplexity Estimates

The working hypothesis is that using the name partition obtained with the (p, γ) procedure we can effectively improve the accuracy of a CDC system by reducing/increasing the amount of contextual evidence required for coreferencing according to the perplexity class to each the name belong.

To construct a test corpus we have adopted the following strategy: we chose 20 two-token names such that both sets of one token-names, the first names and the last names respectively, cover the whole space in the perplexity partition. In Table 6a and 6b we present 5 first and last names used in test. As not all the 25 names formed by combin-

ing the names in 6a and 6b are found in the corpus, we consider 11 other two-tokens names having the same distribution. On the first column the names are listed, on the second column the computed perplexity (P), on the third column the number of occurrences as one-token name (O), on the fourth the number of occurrences in a two-token name (T) and on the last column the computed perplexity class (PC).

Name	P	O	T	PC
Dellai	7	31091	10722	VL
Parolari	171	1,619	2207	H
Prodi	52	9184	3382	M
Ruini	15	554	203	L
Rossi	753	7506	8356	VH

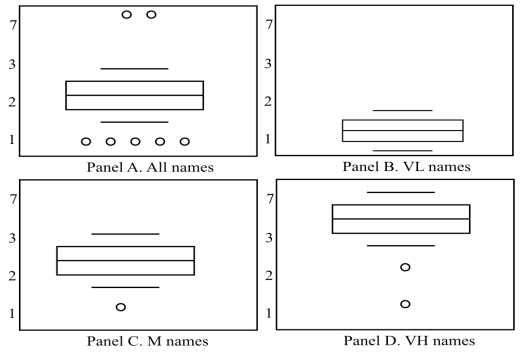
Table 6a. Test Last Names

Name	P	O	T	PC
Camillo	276	664	1731	L
Lorenzo	2088	2167	2198	H
Paolo	5255	4001	51244	VH
Romano	14	886	6414	M
Varena	5	10	85	VH

Table 6b. Test First Names

We compare the results obtained by our CDC system using the name perplexity partition (S) against two baselines: one that considers only the context at the sentence level and (BLS) one that considers the whole news (BLN). We obtain the following figures using the B-CUBED measure: S scores .72, BLS .59 and BLN .61. The gain in accuracy of more than 10% is due to the use of name perplexity classes.

The great advantage of using the (p,γ) estimates can be seen in those case where the ratio between the number of mentions and the rank of the name is close to extremes: either big number of mentions and low name perplexity, or low number of mentions and high name perplexity. In the first case the contextual evidence for coreference may be very scarce and in the second case, the requirement for strong contextual evidence is the best decision. Our results suggest that loosening the contextual requirements in the first case leads to an important gain in recall, up to 40%, while the lose in precision is less than 1.5%. The situation is best described by four panels of the five-number-summary plots of the test corpus. Panel A shows the distribution of the main five quantilies considering all the names together. Panel B shows the distribution for very low perplexity class, Panel C for medium perplexity class and Panel D for the very high perplexity class. The number of outliers in Panel A is high, which makes it difficult for any CDC system, but inside each perplexity class the variation is reduced.



7 Constructing an Evaluation Corpus

The (p, γ) technique could be used for constructing a test corpus for the CDC task. The main problem faced in the construction of the test corpus is data variation. The number of different entities mentioned with the same name is a random variable with a big variance. The distribution of the number of entities is very skew. The average perplexity is 2.01%, but less than 18% of the total number of names have a perplexity greater than 3. In Figure 2 we plot a modified Lorenz curve (the vertical axis is not divided in percentage, as the values are discrete).

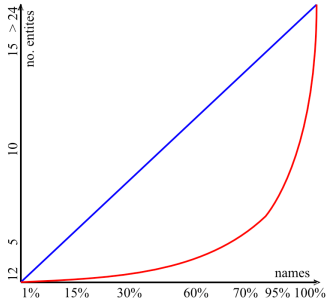


Figure 2. Lorenz Curve names/no. entities

The direct consequence of this situation is the fact that constructing an evaluation corpus by taking random names will result with a great probability in a very skew test corpus. Indeed,

the expectation is that in such corpus, the average perplexity is very low, and consequently, the great majority of cases can be coreferenced by a simple algorithm. Therefore, this test corpus may be largely ineffective in ranking the algorithms. In fact, we want to construct an evaluation corpus that is able to promote the most effective algorithms. The discriminative power of a test corpus is directly related to the variance of the data. Moreover, if only certain names are considered for a test corpus, the variance can be very low; in particular, when the test corpus contains just one name the variance is zero. It is difficult to see the merits of different algorithms when tested on such corpora.

In order to make more informative statements we need to construct an evaluation corpus that is less dependent on the data variance. A possible solution is to form a partition of the set of the PNMs, that is, to split the whole set of PNMs in mutual disjunctive groups. This type of methodology is called stratified sampling, mainly because each group is a stratum. The sampling strategy, the number of sampling elements, the variance and the sampling error can be calculated independently for each strata.

The main advantages of stratified sampling are that we can concentrate on the special groups, that in general this strategy improves the accuracy of the estimation, and that the number of elements in each stratum can be conveniently chosen. The main disadvantages are related to the difficulty in finding a suitable partition of the population. The strata should be chosen prior to the sampling time, but the homogeneity inside the stratum should be guaranteed.

Our proposal is to use the name perplexity intervals. We argue that this proposal is four-fold sustainable. Firstly, the name perplexity is directly connected to the random variable whose distribution we estimate, namely the number of entities. Secondly, for free names it can be computed off - line. Thirdly, it gives us an independent and formally correct way to make a partition. Fourthly, it easily allows a separation between the important and unimportant cases.

To begin with, let us suppose we have a name that has *n* occurrences in the Adige 500K. If *n* is relatively large, than we can be sure that there are some dominant entities that may be represented by the majority of PNMs that have this name as value. However, it is unknown whether the *n* comes from the fact that there are indeed some dominant entities or whether the name by itself is a frequently used name.

In order to deal with the differences between frequency vs. perplexity, we propose to build a matrix defined by the frequency classes as rows and perplexity classes as columns. In Figure 3 we present this matrix.

Frequence/Commoness	VF	F	C	R	VR
Int1					
Int2					
Int3					
Int4					
Int5					

Int1 , Int2, ..., Int5 - frequency
very frequent (VF), ..., common (C), ..., very rare(VR) - perplexity

Figure 3. Frequency/Commonness strata matrix.

The number of different names in each of the cells of the matrix may differ according to the departure of the normal distribution of each stratum. In general, if the real distribution is normal, then as much as ten examples are sufficient. Otherwise, for not very skew distributions, which we expect most of the strata to have, an average of 30 examples should suffice. In same cases, as the normal distribution can be appropriately sampled when both *Np* and *N(1-p)* are grater than five – where p is the ratio perplexity/frequency and N the sample dimension – the number of elements in the cell may be around 200, by a maximal rough estimation.

8 Conclusion and Further Research

We have presented a distributional free statistical method to design a name perplexity system, such that each perplexity class maximizes the number of names for which the prior coreference probability belongs to the same interval. This information helps the PCDC systems lower/increase adequately the amount of contextual evidence required for coreference.

The approach presented here is effective in dealing with the problems raised by using a similarity metrics on contextual vectors improving the overall accuracy with more than 10%.

We would like to increase the number of cases considered in the sample required to delimit the perplexity classes. Equation (3) may be developed further in order to obtain exactly the number of required cases.

The (p, γ) procedure is effective is dealing with the problems regarding the construction of an evaluation corpus. The technique presented in the last section could be extended further and we are already working on a new series of experiments whose results will be made available in the near future.

Acknowledgments

The corpus used in this paper is Adige500k, a seven-year news corpus from an Italian local newspaper. The author thanks to all the people involved in the construction of Adige500k.

References

J. Artiles, Gonzalo, J., S. Sekine. 2007. *Establishing a benchmark for WePS*. In Proceedings of SemEval.

A. Bagga, B. Baldwin. 1998. *Entity-based Cross-Document Co-referencing using the Vector Space Model*. In Proceedings of ACL.

J. Chen, D. Ji, C. Tan, Z. Niu. 2006. *Unsupervised Relation Disambiguation Using Spectral Clustering.* In Proceedings of COLING

C. Gooi, J. Allan. 2004. *Cross-Document Coreference on a Large Scale Corpus*. In Proceedings of ACL.

R. Grishman. 1994. *Whither Written Language Evaluation?* In Proceedings of Human Language Technology Workshop, pp. 120-125. San Mateor.

E. Elmacioglu, Y. M. F. M.Y.Khan, D. Lee. 2007. *PSNUS: Web People Name Disambiguation by Simple Clustering with Rich Features,* in Proceedings of SemEval

H. Han, W. Xu. 2005. *A Hierarchical Bayes Mixture Model for Name Disambiguation in Author Citations*, in Proceedings of SAC'05

R. Hog, J. McKean, A. Craig, 2006. *Introduction of Mathematical Statistics*, ed. Prentice Hall

E. Lefever, V. Hoste, F. Timur. 2007. *AUG: A Combined Classification and Clustering Approach for Web People Disambiguation*, In Proceedings of SemEval

B. Magnini, M. Speranza, M. Negri, L. Romano, R. Sprugnoli. 2006. *I-CAB – the Italian Content Annotation Bank*. LREC 2006

V., Ng. 2007. *Shallow Semantics for Coreference Resolution*, In Proceedings of IJCAI

T. Pedersen, A. Purandare, A. Kulkarni. 2005. *Name Discrimination by Clustering Similar Contexts*, in Proceeding of CICLING

O. Popescu, C. Girardi. 2008. *Improving Cross Document Coreference*, in Proceedings of JADT

O. Popescu, B. Magnini. 2007. *Inferring Coreference among Person Names in a Large Corpus of News Collection*, in Proceedings of AIIA

Y. Wei, M. Lin, H. Chen. 2006. *Name Disambiguation in Person Information Mining,* in Proceedings of IEEE

Q. Vu, T. Massada, A. Takasu, J. Adachi. 2007. *Using Knowledge Base to Disambiguate Personal names in Web Search Results*, In Proceedings of SAC

Learning Linear Ordering Problems for Better Translation*

Roy Tromble
Google, Inc.
4720 Forbes Ave.
Pittsburgh, PA 15213
royt@google.com

Jason Eisner
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
jason@cs.jhu.edu

Abstract

We apply machine learning to the Linear Ordering Problem in order to learn sentence-specific reordering models for machine translation. We demonstrate that even when these models are used as a mere preprocessing step for German-English translation, they significantly outperform Moses' integrated lexicalized reordering model.

Our models are trained on automatically aligned bitext. Their form is simple but novel. They assess, based on features of the input sentence, how strongly each pair of input word tokens w_i, w_j would like to reverse their relative order. Combining all these pairwise preferences to find the best global reordering is NP-hard. However, we present a non-trivial $O(n^3)$ algorithm, based on chart parsing, that at least finds the best reordering within a certain exponentially large neighborhood. We show how to iterate this reordering process within a local search algorithm, which we use in training.

1 Introduction

Machine translation is an important but difficult problem. One of the properties that makes it difficult is the fact that different languages express the same concepts in different orders. A machine translation system must therefore rearrange the source language concepts to produce a fluent translation in the target language.

Phrase-based translation systems rely heavily on the target language model to ensure a fluent output order. However, a target n-gram language model alone is known to be inadequate. Thus, translation systems should also look at *how the source sentence prefers to reorder*. Yet past systems have traditionally used rather weak models of the reordering process. They may look only at the distance between neighboring phrases, or depend only on phrase unigrams. The decoders also rely on search error, in the form of limited reordering windows, for both efficiency and translation quality.

Demonstrating the inadequacy of such approaches, Al-Onaizan and Papineni (2006) showed that even *given* the words in the reference translation, *and* their alignment to the source words, a decoder of this sort charged with merely rearranging them into the correct target-language order could achieve a BLEU score (Papineni et al., 2002) of at best 69%—and that only when restricted to keep most words very close to their source positions.

This paper introduces a more sophisticated model of reordering based on the Linear Ordering Problem (LOP), itself an NP-hard permutation problem. We apply machine learning, in the form of a modified perceptron algorithm, to learn parameters of a linear model that constructs a matrix of weights from each source language sentence. We train the parameters on orderings derived from automatic word alignments of parallel sentences.

The LOP model of reordering is a complete ordering model, capable of assigning a different score to every possible permutation of the source-language sentence. Unlike the target language model, it uses information about the relative positions of the words in the source language, as well as the source words themselves and their parts of speech and contexts. It is therefore a language-pair specific model.

[1]This work is excerpted and adapted from the first author's Ph.D. thesis (Tromble, 2009). Some of the ideas here appeared in (Eisner and Tromble, 2006) without empirical validation. The material is based in part upon work supported by the National Science Foundation under Grant No. 0347822.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1007–1016,
Singapore, 6-7 August 2009.

We apply the learned LOP model as a preprocessing step before both training and evaluation of a phrase-based translation system, namely Moses. Our methods for finding a good reordering under the NP-hard LOP are themselves of interest, adapting algorithms from natural language parsing and developing novel dynamic programs.

Our results demonstrate a significant improvement over translation using unreordered German. Using Moses with only distance-based reordering and a distortion limit of 6, our preprocessing improves BLEU from 25.27 to 26.40. Furthermore, that improvement is significantly greater than the improvement Moses achieves with its lexicalized reordering model, 25.55.

Collins et al. (2005) improved German-English translation using a statistical parser and several hand-written rules for preprocessing the German sentences. This paper presents a similar improvement using fully automatic methods.

2 A Linear Ordering Model

This section introduces a model of word reordering for machine translation based on the Linear Ordering Problem.

2.1 Formalization

The input sentence is $\boldsymbol{w} = w_1 w_2 \ldots w_n$. To distinguish duplicate tokens of the same word, we assume that each token is superscripted by its input position, e.g., $\boldsymbol{w} =$ *die*^{1} *Katze*^{2} *hat*^{3} *die*^{4} *Frau*^{5} *gekauft*^{6} (gloss: "the cat has the woman bought").

For a fixed $\boldsymbol{w}$, a permutation $\boldsymbol{\pi} = \pi_1 \pi_2 \ldots \pi_n$ is any reordering of the tokens in $\boldsymbol{w}$. The set Π_n of all such permutations has size $n!$. We would like to define a scoring model that assigns a high score to the permutation $\boldsymbol{\pi} =$ *die*^{4} *Frau*^{5} *hat*^{3} *gekauft*^{6} *die*^{1} *Katze*^{2} (gloss: "the woman has bought the cat"), since that corresponds well to the desired English order.

To construct a function that scores permutations of $\boldsymbol{w}$, we first construct a pairwise preference matrix $B_{\boldsymbol{w}} \in \mathbb{R}^{n \times n}$, whose entries are

$$B_{\boldsymbol{w}}[\ell, r] \stackrel{\text{def}}{=} \boldsymbol{\theta} \cdot \boldsymbol{\phi}(\boldsymbol{w}, \ell, r), \qquad (1)$$

Here $\boldsymbol{\theta}$ is a vector of weights. $\boldsymbol{\phi}$ is a vector of feature functions, each considering the entire word sequence $\boldsymbol{w}$, as well as any functions thereof, such as part of speech tags.

We will hereafter abbreviate $B_{\boldsymbol{w}}$ as B. Its integer indices ℓ and r are identified with the input tokens w_ℓ and w_r, and it can be helpful to write them that way; e.g., we will sometimes write $B[2, 5]$ as $B[\textit{Katze}^2, \textit{Frau}^5]$.

The idea behind our reordering model is that $B[\textit{Katze}^2, \textit{Frau}^5] > B[\textit{Katze}^5, \textit{Frau}^2]$ expresses a preference to keep *Katze*^{2} before *Frau*^{5}, whereas the opposite inequality would express a preference—other things equal—for permutations in which their order is reversed. Thus, we define[1]

$$\text{score}(\boldsymbol{\pi}) \stackrel{\text{def}}{=} \sum_{i,j:\ 1 \le i < j \le n} B[\pi_i, \pi_j] \qquad (2)$$

$$p(\boldsymbol{\pi}) \stackrel{\text{def}}{=} \frac{1}{Z} \exp(\gamma \cdot \text{score}(\boldsymbol{\pi})) \qquad (3)$$

$$\hat{\boldsymbol{\pi}} \stackrel{\text{def}}{=} \operatorname*{argmax}_{\boldsymbol{\pi} \in \Pi_n} \text{score}(\boldsymbol{\pi}) \qquad (4)$$

Note that i and j denote positions in $\boldsymbol{\pi}$, whereas π_i, π_j, ℓ, and r denote particular input tokens such as *Katze*^{2} and *Frau*^{5}.

2.2 Discussion

To the extent that the costs B generally discourage reordering, they will particularly discourage long-distance movement, as it swaps more pairs of words.

We point out that our model is somewhat peculiar, since it does not directly consider whether the permutation $\boldsymbol{\pi}$ keeps *die*^{4} and *Frau*^{5} adjacent or even close together, but only whether their order is reversed.

Of course, the model could be extended to consider adjacency, or more generally, the three-way cost of interposing k between i and j. See (Eisner and Tromble, 2006; Tromble, 2009) for such extensions and associated algorithms.

However, in the present paper we focus on the model in the simple form (2) that only considers pairwise reordering costs for all pairs in the sentence. Our goal is to show that these unfamiliar pairwise reordering costs are useful, when modeled with a rich feature set via equation (1). Even *in isolation* (as a preprocessing step), without considering any other kinds of reordering costs or language model, they can achieve useful reorderings

[1] For any $\ell < r$, we may assume without loss of generality that $B[r, \ell] = 0$, since if not, subtracting $B[r, \ell]$ from both $B[\ell, r]$ and $B[r, \ell]$ (exactly one of which appears in each $\text{score}(\boldsymbol{\pi})$) will merely reduce the scores of *all* permutations by this amount, leaving equations (3) and (4) unchanged. Thus, in practice, we take B to be an upper triangular matrix. We use equation (1) only to define $B[\ell, r]$ for $\ell < r$, and train $\boldsymbol{\theta}$ accordingly. However, we will ignore this point in our exposition.

of German that complement existing techniques and thus improve state-of-the-art systems. Our positive results in even this situation suggest that in future, pairwise reordering costs should probably be integrated into MT systems.

The probabilistic interpretation (3) of the score (2) may be useful when thus integrating our model with language models or other reordering models during translation, or simply when training our model to maximize likelihood or minimize expected error. However, in the present paper we will stick to purely discriminative training and decoding methods that simply try to maximize (2).

2.3 The Linear Ordering Problem

In the combinatorial optimization literature, the maximization problem (4) (with input B) is known as the Linear Ordering Problem. It has numerous practical applications in fields including economics, sociology, graph theory, graph drawing, archaeology, and task scheduling (Grötschel et al., 1984). Computational studies on real data have often used "input-output" matrices representing resource flows among economic sectors (Schiavinotto and Stützle, 2004).

Unfortunately, the problem is NP-hard. Furthermore, it is known to be APX-complete, meaning that there is no polynomial time approximation scheme unless P=NP (Mishra and Sikdar, 2004). However, there are various heuristic procedures for approximating it (Tromble, 2009). We now give an attractive, novel procedure, which uses a CKY-parsing-like algorithm to search various subsets of Π_n in polynomial time.

3 Local Search

"Local search" refers to any hill-climbing procedure that iteratively improves a solution by making an optimal "local" change at each iteration.[2] In this case, we start with the identity permutation, find a "nearby" permutation with a better score (2), and repeat until we have reached a local maximum of the scoring objective.

This section describes a local search procedure that uses a very generous definition of "local." At each iteration, it finds the optimal permutation in a certain *exponentially large* neighborhood $N(\boldsymbol{\pi})$ of the current permutation $\boldsymbol{\pi}$.

[2]One can introduce randomness to obtain MCMC sampling or simulated annealing algorithms. Our algorithms extend naturally to allow this (cf. Tromble (2009)).

$$\begin{aligned} S &\rightarrow S_{0,n} \\ S_{i,k} &\rightarrow S_{i,j}\ S_{j,k} \\ S_{i-1,i} &\rightarrow \pi_i \end{aligned}$$

Figure 1: A grammar for a large neighborhood of permutations, given one permutation $\boldsymbol{\pi}$ of length n. The $S_{i,k}$ rules are instantiated for each $0 \leq i < j < k \leq n$, and the $S_{i-1,i}$ rules for each $0 < i \leq n$.

We say that two permutations are neighbors iff they can be aligned by an Inversion Transduction Grammar (ITG) (Wu, 1997), which is a familiar reordering device in machine translation. Equivalently, $\boldsymbol{\pi}' \in N(\boldsymbol{\pi})$ iff $\boldsymbol{\pi}$ can be transformed into $\boldsymbol{\pi}'$ by swapping various adjacent substrings of $\boldsymbol{\pi}$, as long as these swaps are properly nested. Zens and Ney (2003) used a normal form to show that the size of the ITG neighborhood $N(\boldsymbol{\pi})$ is a large Schröder number, which grows exponentially in n. Asymptotically, the ratio between the size of the neighborhood for $n+1$ and the size for n approaches $3 + 2\sqrt{2} \approx 5.8$.

We show that equation (2) can be optimized within $N(\boldsymbol{\pi})$ in $O(n^3)$ time, using dynamic programming. The algorithm is based on CKY parsing. However, a novelty is that the grammar weights must themselves be computed by $O(n^3)$ dynamic programming.

Our grammar is shown in Figure 1. Parsing the "input sentence" $\boldsymbol{\pi}$ with this grammar simply constructs all binary trees that yield the string $\boldsymbol{\pi}$. There is essentially only one nonterminal, S, but we split it into $O(n^2)$ position-specific nonterminals such as $S_{i,j}$, which can only yield the span $\pi_{i+1}\pi_{i+2}\ldots\pi_j$. An example parse is shown in Figure 2.

The important point is that we will place a score on each binary grammar rule. The score of the rule $S_{i,k} \rightarrow S_{i,j}\ S_{j,k}$ is $\max(0, \Delta_{i,j,k})$, where $\Delta_{i,j,k}$ is the benefit to swapping the substrings $\pi_{i+1}\pi_{i+2}\ldots\pi_j$ and $\pi_{j+1}\pi_{j+2}\ldots\pi_k$. The rule is considered to be a "swap rule" if its score is positive, showing that a swap will be beneficial (independent of the rest of the tree). If the parse in Figure 2 is the parse with the highest *total* score, and its swap rules are $S_{0,5} \rightarrow S_{0,1}\ S_{1,5}$ and $S_{3,5} \rightarrow S_{3,4}\ S_{4,5}$, then our best permutation in the neighborhood of $\boldsymbol{\pi}$ must be the (linguistically desirable) permutation *die*^{4}*Frau*^{5}*hat*^{3}*gekauft*^{6}*die*^{1}*Katze*^{2}, obtained from

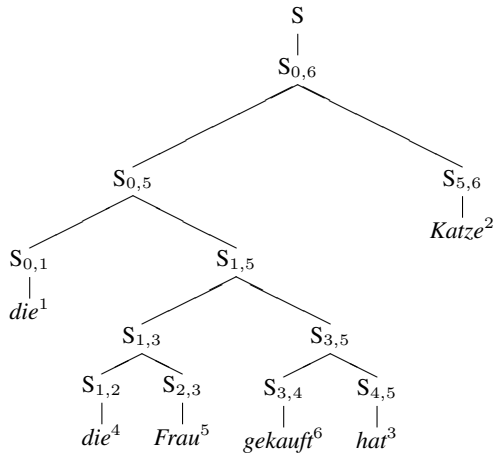


Figure 2: One parse of the current permutation $\boldsymbol{\pi}$. In this example, $\boldsymbol{\pi}$ has somehow gotten the input words into alphabetical order (owing to previous hill-climbing steps). We are now trying to further improve this order.

$\boldsymbol{\pi}$ by two swaps.

How do we find this solution? Clearly the benefit (positive or negative) to swapping $\pi_{i+1}\pi_{i+2}\ldots\pi_j$ with $\pi_{j+1}\pi_{j+2}\ldots\pi_k$ is

$$\Delta_{i,j,k} = \sum_{\ell=i+1}^{j}\sum_{r=j+1}^{k} B[\pi_r, \pi_\ell] - B[\pi_\ell, \pi_r] \quad (5)$$

We can evaluate all $O(n^3)$ possible swaps in total time $O(n^3)$, using the dynamic programming recurrence

$$\begin{aligned}\Delta_{i,j,k} &= \Delta_{i,j,k-1} + \Delta_{i+1,j,k} - \Delta_{i+1,j,k-1} \quad (6)\\ &\quad + B[\pi_k, \pi_{i+1}] - B[\pi_{i+1}, \pi_k]\end{aligned}$$

with the base case $\Delta_{i,j,k} = 0$ if $i = j$ or $j = k$. This gives us the weights for the grammar rules, and then we can use weighted CKY parsing to find the highest-scoring (Viterbi) parse in $O(n^3)$ time. Extracting our new and improved permutation $\boldsymbol{\pi'} \in N(\boldsymbol{\pi})$ from this parse is a simple $O(n)$-time algorithm.

Figure 3 gives pseudocode for our local search algorithm, showing how to compute the quantities (6) during parsing rather than beforehand. $\beta[i,k]$ holds the weight of the best permutation (in the neighborhood) of the subsequence $\pi_{i+1}\pi_{i+1}\ldots\pi_k$.[3]

[3]The use of β is intended to suggest an analogy to inside probability—or more precisely, the Viterbi approximation to inside probability (since we are maximizing rather than summing over parses).

The next two sections describe how to use our local search algorithm to discriminatively learn the weights of the parameters from Section 2, equation (1).

4 Features

Our objective function (2) works only to the extent that we can derive a good pairwise preference matrix $B_{\boldsymbol{w}}$. We do this by using a rich feature set in equation (1).

We adapt the features of McDonald et al. (2005), introduced there for dependency parsing, to the task of machine translation reordering. Because both models construct features for pairs of words given the entire sentence, there is a close correspondence between the two tasks, although the output is quite different.

Each feature $\phi(\boldsymbol{w}, \ell, r)$ in equation (1) is a binary feature that fires when $(\boldsymbol{w}, \ell, r)$ has some conjunction of properties. The properties that are considered include the words w_ℓ and w_r, the parts of speech of $\{w_{\ell-1}, \ldots, w_{r+1}\}$, and the distance $r - \ell$. Table 1 shows the feature templates.

We also tried features based on a dependency parse of the German, with the idea of using LOP features to reorder the dependents of each word, and thus model syntactic movement. This did produce better monolingual reorderings (as in Table 2), but it did not help final translation into English (Table 3), so we do not report the details here.

5 Learning to Reorder

Ideally, we would have a large corpus of desirable reorderings of source sentences—in our case, German sentences permuted into target English word order—from which to train the parameters of our model. Unfortunately, the alignments between German and English sentences are only infrequently one-to-one. Furthermore, human-aligned parallel sentences are hard to come by, and never in the quantity we would like.

Instead, we make do with automatically-generated word alignments, and we heuristically derive an English-like word order for the German sentence based on the alignment. We used GIZA++ (Och and Ney, 2003) to align approximately 751,000 sentences from the German-English portion of the Europarl corpus (Koehn, 2005), in both the German-to-English and English-to-German directions. We combined the

```
1: procedure LocalSearchStep(B, π, n)
2:     for i ← 0 to n − 1 do
3:         β[i, i + 1] ← 0
4:         for k ← i + 1 to n do
5:             Δ[i, i, k] ← Δ[i, k, k] ← 0
6:         end for
7:     end for
8:     for w ← 2 to n do
9:         for i ← 0 to n − w do
10:            k ← i + w
11:            β[i, k] ← −∞
12:            for j ← i + 1 to k − 1 do
13:                Δ[i, j, k] ← Δ[i, j, k − 1] + Δ[i + 1, j, k] − Δ[i + 1, j, k − 1] + B[π_k, π_{i+1}] − B[π_{i+1}, π_k]
14:                β[i, k] ← max(β[i, k], β[i, j] + β[j, k] + max(0, Δ[i, j, k]))
15:            end for
16:        end for
17:    end for
18:    return β[0, n]
19: end procedure
```

Figure 3: Pseudocode for computing the score of the best permutation in the neighborhood of $\boldsymbol{\pi}$ under the Linear Ordering Problem specified by the matrix B. Computing the best neighbor is a simple matter of keeping back pointers to the choices of $\max$ and ordering them as implied.

alignments using the "grow-diag-final-and" procedure provided with Moses (Koehn et al., 2007).

For each of these German sentences, we derived the English-like reordering of it, which we call German$'$, by the following procedure. Each German token was assigned an integer key, namely the position of the leftmost of the English tokens to which it was aligned, or 0 if it was not aligned to any English tokens. We then did a stable sort of the German tokens based on these keys, meaning that if two German tokens had the same key, their order was preserved.

This is similar to the oracle ordering used by Al-Onaizan and Papineni (2006), but differs in the handling of unaligned words. They kept unaligned words with the closest preceding aligned word.[4]

Having found the German$'$ corresponding to each German sentence, we randomly divided the sentences into 2,000 each for development and evaluation, and the remaining approximately 747,000 for training.

We used the averaged perceptron algorithm (Freund and Schapire, 1998; Collins, 2002) to train the parameters of the model. We ran the algorithm multiple times over the training sentences, measuring the quality of the learned parameters by reordering the held-out development set after each iteration. We stopped when the BLEU score on the development set failed to improve for two consecutive iterations, which occurred after fourteen passes over the data.

Each perceptron update should compare the true German$'$ to the German$'$ that would be predicted by the model (2). As the latter is NP-hard to find, we instead substitute the local maximum found by local search as described in Section 3, starting at the identity permutation, which corresponds to the original German word order.

During training, we iterate the local search as described earlier. However, for decoding, we only do a single step of local search, thus restricting reorderings to the ITG neighborhood of the original German. This restriction turns out to improve performance slightly, even though it reduces the quality of our approximation to the LOP problem (4). In other words, it turns out that reorderings found outside the ITG neighborhood tend to be poor German$'$ even if our LOP-based objective function thinks that they are good German$'$.

This is not to say that the gold standard German$'$ is always in the ITG neighborhood of the original German—often it is not. Thus, it might be better in future work to still allow the local search to take more than one step, but to penalize the second step. In effect, score($\boldsymbol{\pi}$) would then include a feature indicating whether $\boldsymbol{\pi}$ is in the neighborhood of the original German.

[4] We tried two other methods for deriving English word order from word alignments. The first alternative was to align only in one direction, from English to German, with null alignments disallowed, so that every German word was aligned to a single English word. The second alternative used BerkeleyAligner (Liang et al., 2006; DeNero and Klein, 2007), which shares information between the two alignment directions to improve alignment quality. Neither alternative produced improvements in our ultimate translation quality.

$t_{\ell-1}$	w_ℓ	t_ℓ	$t_{\ell+1}$	t_b	t_{r-1}	w_r	t_r	t_{r+1}
	×	×				×	×	
	×	×				×		
	×					×	×	
	×	×					×	
		×				×	×	
	×					×		
		×					×	
	×	×						
						×	×	
	×							
						×		
		×						
							×	
		×		×			×	
		×	×		×		×	
		×	×				×	
		×	×				×	×
		×					×	×
×		×					×	×
×		×					×	
×		×			×		×	
		×			×		×	

Table 1: Feature templates for $B[\ell, r]$ (w_ℓ is the ℓth word, t_ℓ its part of speech tag, and b matches any index such that $\ell < b < r$). Each of the above is also conjoined with the distance between the words, $r - \ell$, to form an additional feature template. Distances are binned into 1, 2, 3, 4, 5, > 5, and > 10.

The model is initialized at the start of training using log-odds of the parameters. Let $\Phi_m = \{(\boldsymbol{w}, \ell, r) \mid \phi_m(\boldsymbol{w}, \ell, r) = 1\}$ be the set of word pairs in the training data for which feature m fires. Let $\vec{\Phi}_m$ be the subset of Φ_m for which the words stay in order, and $\overleftarrow{\Phi}_m$ the subset for which the words reverse order. Then in this model,

$$\theta_m = \log\left(\left|\vec{\Phi}_m\right| + \frac{1}{2}\right) - \log\left(\left|\overleftarrow{\Phi}_m\right| + \frac{1}{2}\right). \quad (7)$$

This model is equivalent to smoothed naïve Bayes if converted to probabilities. The learned model significantly outperforms it on the monolingual reordering task.

Table 2 compares the model after perceptron training to the model at the start of training, measuring BLEU score of the predicted German′ against the observed German′. In addition to these BLEU scores, we can measure precision and recall of pairs of reordered words against the observed German′. On the held out test set, the predicted German′ achieves a recall of only 21%, but a precision of 64%. Thus, the learned model is too conservative, but makes moderately good decisions when it does reorder.

Ordering	p_2	p_3	p_4	**BLEU**
German	57.4	38.3	27.7	49.65
Log-odds	57.4	38.4	27.8	49.75
Perceptron	**58.6**	**40.3**	**29.8**	**51.51**

Table 2: Monolingual BLEU score on development data, measured against the "true" German′ ordering that was derived from automatic alignments to known English translations. The table evaluates three candidate orderings: the original German, German reordered using the log-odds initialized model, and German reordered using the perceptron-learned model. In addition to the BLEU score, the table shows bigram, trigram, and 4-gram precisions. The unigram precisions are always 100%, because the correct words are given.

6 Reordering as Preprocessing

This section describes experiments using the model introduced in Section 2 and learned in Section 5 to preprocess German sentences for translation into English. These experiments are similar to those of Collins et al. (2005).

We used the model learned in Section 5 to generate a German′ ordering of the training, development, and test sets. The training sentences are the same that the model was trained on, and the development set is the same that was used as the stopping criterion for the perceptron. The test set was unused in training.

We used the resulting German′ as the input to the Moses training pipeline. That is, Moses recomputed alignments of the German′ training data to the English sentences using GIZA++, then constructed a phrase table. Moses used the development data for minimum error-rate training (Och, 2003) of its small number of parameters. Finally, Moses translated the test sentences, and we measured performance against the English reference sentences. This is the standard Moses pipeline, except German has been replaced by German′.

Table 3 shows the results of translation, both starting with unreordered German, and starting with German′, reordered using the learned Linear Ordering Problems. Note that Moses may itself re-

System	Input	Moses Reord.	p_1	p_2	p_3	p_4	BLEU	METEOR	TER
baseline	German	Distance	59.6	31.4	18.8	11.6	25.27	54.03	60.60
(a)	German	Lexical	60.0	32.0	19.3	12.1	25.55	54.18	59.76
(b)	German′	Distance	60.4	32.7	20.2	12.8	26.40	**54.91**	**58.63**
(a)+(b)	German′	Lexical	59.9	32.4	20.0	12.8	**26.44**	54.61	59.23

Table 3: Machine translation performance of several systems, measured against a single English reference translation. The results vary both the preprocessing—either none, or reordered using the learned Linear Ordering Problems—and the reordering model used in Moses. Performance is measured using BLEU, METEOR (Lavie et al., 2004), and TER (Snover et al., 2006). (For TER, smaller values are better.)

order whatever input that it receives, during translation into English. Thus, the results in the table also vary the reordering model used in Moses, set to either a single-parameter distance-based model, or to the lexicalized bidirectional msd model. The latter model has six parameters for each phrase in the phrase table, corresponding to monotone, swapped, or discontinuous ordering relative to the previous phrase in either the source or target language.

How should we understand the results? The baseline system is Moses phrase-based translation with no preprocessing and only a simple distance-based reordering model. There are two ways to improve this: (a) ask Moses to use the lexicalized bidirectional msd reordering model that is provided with Moses and is integrated with the rest of translation, or (b) keep the simple distance-based model within Moses, but preprocess its training and test data with our linear reordering model. Note that the preprocessing in (b) will obviously change the phrasal substrings that are learned by Moses, for better or for worse.

First, remarkably, (b) is significantly better than (a) on BLEU, with $p < 0.0001$ according to a paired permutation test.

Second, combining (a) with (b) produced no improvement over (b) in BLEU score (the difference between 26.40 and 26.44 is not significant, even at $p < 0.2$, according to the same paired permutation test). Lexicalized reordering in Moses even degraded translation performance according to METEOR and TER. The TER change is significant according to the paired permutation test at $p < 0.001$. (We did not perform a significance test for METEOR.)

Our word-based model surpasses the lexicalized reordering in Moses largely because of long-distance movement. The 518 sentences (26%) in the test set for which our model moves a word more than six words away from its starting position account for more than 67% of the improvement in BLEU from (a) to (b).

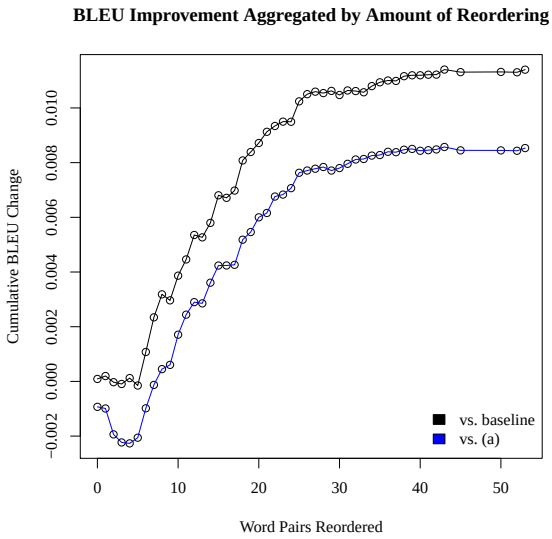


Figure 4: Cumulative change in BLEU score of (b) relative to the baseline and (a), aggregated by the number of reordered word pairs in each sentence. For those sentences where our model reorders fewer than five word pairs, the BLEU score of translation degrades.

Figure 4 shows another view of the BLEU improvement. It shows that, compared to the baseline, our preprocessing has basically no effect for sentences where it does only a little reordering, changing the relative order of fewer than five pairs of words. Compared to Moses with lexicalized reordering, these same sentences actually hurt performance. This more than accounts for the difference between the BLEU scores of (b) and (a)+(b).

Going beyond preprocessing, our model could also be integrated into a phrase-based decoder. We briefly sketch that possibility here.

Phrase-based decoders keep a source coverage vector with every partial translation hypothesis. That coverage vector allows us to incorporate the scores from a LOP matrix B directly. Whenever the decoder extends the hypothesis with a new source phrase, covering $w_{i+1}w_{i+2}\ldots w_j$, it adds

$$\sum_{\ell=i+1}^{j-1}\sum_{r=\ell+1}^{j} B[\ell,r] + \sum_{\ell=i+1}^{j}\sum_{r\in\mathcal{U}} B[\ell,r].$$

The first term represents the phrase-internal score, and the second the score of putting the words in the phrase before all the remaining uncovered words $\mathcal{U}$.

7 Comparison to Prior Work

Preprocessing the source language to improve translation is a common technique. Xia and McCord (2004) improved English-French translation using syntactic rewrite rules derived from Slot Grammar parses. Collins et al. (2005) reported an improvement from 25.2% to 26.8% BLEU on German-English translation using six handwritten rules to reorder the German sentences based on automatically-generated phrase-structure trees. Our work differs from these approaches in providing an explicit model that scores all possible reorderings. In this paper, our model was trained and used only for 1-best preprocessing, but it could potentially be integrated into decoding as well, where it would work together with the translation model and target language model to find a congenial translation.

Costa-jussà and Fonollosa (2006) improved Spanish-English and Chinese-English translation using a two-step process, first reordering the source language, then translating it, both using different versions of a phrase-based translation system. Many others have proposed more explicit reordering models (Tillmann, 2004; Kumar and Byrne, 2005; Koehn et al., 2005; Al-Onaizan and Papineni, 2006). The primary advantage of our model is that it directly accounts for interactions between distant words, leading to better treatment of long-distance movement.

Xiong et al. (2006) proposed a constituent reordering model for a bracketing transduction grammar (BTG) (Wu, 1995), which predicts the probability that a pair of subconstituents will reorder when combined to form a new constituent. The features of their model look only at the *first* source and target word of each constituent, making it something like a sparse version of our model. However, because of the target word features, their reordering model cannot be separated from their translation model.

8 Conclusions and Future Work

We have presented an entirely new model of reordering for statistical machine translation, based on the Linear Ordering Problem, and shown that it can substantially improve translation from German to English.

The model is demonstrably useful in this preprocessing setting—which means that it can be very simply added as a preprocessing step to any MT system. German-to-English is a particularly attractive use case, because the word orders are sufficiently different as to require a good reordering model that requires long-distance reordering. Our preprocessing here gave us a BLEU gain of 0.9 point over the best Moses-based result. English-to-German would obviously be another potential win, as would translating between English and Japanese, for example.

As mentioned in Section 6, our model could also be integrated into a phrase-based, or a syntax-based decoder. That possibility remains future work, but it is likely to lead to further improvements, because it allows the translation system to consider multiple possible reorderings under the model, as well as to tune the weight of the model relative to the other parts of the system during MERT.

Tromble (2009) covers this integration in more detail, and proposes several other ways of integrating our reordering model into machine translation. It also experiments with numerous other parameter estimation procedures, including some that use the probabilistic interpretation of our model from (3). It presents numerous additional neighborhoods for search in the Linear Ordering Problem.

We mentioned several possible extensions to the model, such as going beyond the scoring model of equation (2), or considering syntax-based features. Another extension would try to reorder not words but phrases, following (Xiong et al., 2006), or segment choice models (Kuhn et al., 2006), which assume a single segmentation of the words into phrases. We would have to define the pairwise preference matrix B over phrases rather than

words (Eisner and Tromble, 2006). This would have the disadvantage of complicating the feature space, but might be a better fit for integration with a phrase-based decoder.

Finally, we gave a novel algorithm for approximately solving the Linear Ordering Problem, interestingly combining dynamic programming with local search. Another novel contribution is that we showed how to parameterize a function that constructs a specific Linear Ordering Problem instance from an input sentence $\boldsymbol{w}$, and showed how to learn those parameters from a corpus of parallel sentences, using the perceptron algorithm. Likelihood-based training using equation (3) would also be possible, with modifications to our algorithm, notably the use of normal forms to avoid counting some permutations multiple times (Tromble, 2009).

It would be interesting to compare the speed and accuracy of our dynamic-programming local-search method with an exact algorithm for solving the LOP, such as integer linear programming with branch and bound (cf. Charon and Hudry (2006)). Exact solutions can generally be found in practice for $n \leq 100$.

References

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *COLING-ACL*, pages 529–536, Sydney, July.

Irène Charon and Olivier Hudry. 2006. A branch-and-bound algorithm to solve the linear ordering problem for weighted tournaments. *Discrete Applied Mathematics*, 154(15):2097–2116, October.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *ACL*, pages 531–540, Ann Arbor, Michigan, June.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8, Philadelphia, July.

Marta R. Costa-jussà and José A. R. Fonollosa. 2006. Statistical machine reordering. In *EMNLP*, pages 70–76, Sydney, July.

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *ACL*, pages 17–24, Prague, June.

Jason Eisner and Roy W. Tromble. 2006. Local search with very large-scale neighborhoods for optimal permutations in machine translation. In *Workshop on computationally hard problems and joint inference in speech and language processing*, New York, June.

Yoav Freund and Robert E. Schapire. 1998. Large margin classification using the perceptron algorithm. In *COLT*, pages 209–217, New York. ACM Press.

Martin Grötschel, Michael Jünger, and Gerhard Reinelt. 1984. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32(6):1195–1220, November–December.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *IWSLT*, Pittsburgh, October.

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

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit X*, pages 79–86, Phuket, Thailand, September.

Roland Kuhn, Denis Yuen, Michel Simard, Patrick Paul, George Foster, Eric Joanis, and Howard Johnson. 2006. Segment choice models: Feature-rich models for global distortion in statistical machine translation. In *HLT-NAACL*, pages 25–32, New York, June.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *HLT-EMNLP*, pages 161–168, Vancouver, October.

Alon Lavie, Kenji Sagae, and Shyamsundar Jayaraman. 2004. The signicance of recall in automatic metrics for MT evaluation. In Robert E. Frederking and Kathryn B. Taylor, editors, *Machine Translation: From Real Users to Research*, pages 134–143. AMTA, Springer, September–October.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *HLT-NAACL*, pages 104–111, New York, June.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Spanning tree methods for discriminative training of dependency parsers. Technical Report MS-CIS-05-11, UPenn CIS.

Sounaka Mishra and Kripasindhu Sikdar. 2004. On approximability of linear ordering and related NP-optimization problems on graphs. *Discrete Applied Mathematics*, 136(2–3):249–269, February.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*, pages 160–167, Sapporo, July.

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

Tommaso Schiavinotto and Thomas Stützle. 2004. The linear ordering problem: Instances, search space analysis and algorithms. *Journal of Mathematical Modelling and Algorithms*, 3(4):367–402, December.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *AMTA*.

Christoph Tillmann. 2004. A unigram orientation model for statistical machine translation. In *HLT-NAACL Short Papers*, pages 101–104, Boston, May.

Roy Wesley Tromble. 2009. *Search and Learning for the Linear Ordering Problem with an Application to Machine Translation*. Ph.D. thesis, Johns Hopkins University, Baltimore, April. `http://nlp.cs.jhu.edu/~royt/`

Dekai Wu. 1995. An algorithm for simultaneously bracketing parallel texts by aligning words. In *ACL*, pages 244–251, Cambridge, Massachusetts, June.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, September.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *COLING*, pages 508–514, Geneva, August.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *COLING-ACL*, pages 521–528, Sydney, July.

Richard Zens and Hermann Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *ACL*, pages 144–151, Sapporo, July.

Weighted Alignment Matrices for Statistical Machine Translation

Yang Liu , Tian Xia , Xinyan Xiao and **Qun Liu**
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
{yliu,xiatian,xiaoxinyan,liuqun}@ict.ac.cn

Abstract

Current statistical machine translation systems usually extract rules from bilingual corpora annotated with 1-best alignments. They are prone to learn noisy rules due to alignment mistakes. We propose a new structure called *weighted alignment matrix* to encode all possible alignments for a parallel text compactly. The key idea is to assign a probability to each word pair to indicate how well they are aligned. We design new algorithms for extracting phrase pairs from weighted alignment matrices and estimating their probabilities. Our experiments on multiple language pairs show that using weighted matrices achieves consistent improvements over using n-best lists in significant less extraction time.

1 Introduction

Statistical machine translation (SMT) relies heavily on annotated bilingual corpora. Word alignment, which indicates the correspondence between the words in a parallel text, is one of the most important annotations in SMT. Word-aligned corpora have been found to be an excellent source for translation-related knowledge, not only for phrase-based models (Och and Ney, 2004; Koehn et al., 2003), but also for syntax-based models (e.g., (Chiang, 2007; Galley et al., 2006; Shen et al., 2008; Liu et al., 2006)). Och and Ney (2003) indicate that the quality of machine translation output depends directly on the quality of initial word alignment.

Modern alignment methods can be divided into two major categories: *generative* methods and *discriminative* methods. Generative methods (Brown et al., 1993; Vogel and Ney, 1996) treat word alignment as a hidden process and maximize the likelihood of bilingual training corpus using the expectation maximization (EM) algorithm. In contrast, discriminative methods (e.g., (Moore et al., 2006; Taskar et al., 2005; Liu et al., 2005; Blunsom and Cohn, 2006)) have the freedom to define arbitrary feature functions that describe various characteristics of an alignment. They usually optimize feature weights on manually-aligned data. While discriminative methods show superior alignment accuracy in benchmarks, generative methods are still widely used to produce word alignments for large sentence-aligned corpora.

However, neither generative nor discriminative alignment methods are reliable enough to yield high quality alignments for SMT, especially for distantly-related language pairs such as Chinese-English and Arabic-English. The F-measures for Chinese-English and Arabic-English are usually around 80% (Liu et al., 2005) and 70% (Fraser and Marcu, 2007), respectively. As most current SMT systems only use 1-best alignments for extracting rules, alignment errors might impair translation quality.

Recently, several studies have shown that offering more alternatives of annotations to SMT systems will result in significant improvements, such as replacing 1-best trees with packed forests (Mi et al., 2008) and replacing 1-best word segmentations with word lattices (Dyer et al., 2008). Similarly, Venugopal et al. (2008) use n-best alignments instead of 1-best alignments for translation rule extraction. While they achieve significant improvements on the IWSLT data, extracting rules from n-best alignments might be computationally expensive.

In this paper, we propose a new structure named *weighted alignment matrix* to represent the alignment distribution for a sentence pair compactly. In a weighted matrix, each element that corresponds to a word pair is assigned a probability to measure the confidence of aligning the two words. Therefore, a weighted matrix is capable of using a lin-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1017–1026,
Singapore, 6-7 August 2009.

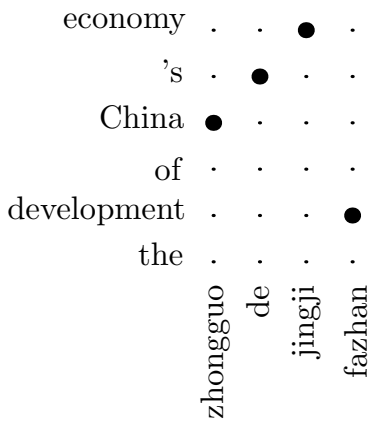


Figure 1: An example of word alignment between a pair of Chinese and English sentences.

ear space to encode the probabilities of exponentially many alignments. We develop a new algorithm for extracting phrase pairs from weighted matrices and show how to estimate their relative frequencies and lexical weights. Experimental results show that using weighted matrices achieves consistent improvements in translation quality and significant reduction in extraction time over using n-best lists.

2 Background

Figure 1 shows an example of word alignment between a pair of Chinese and English sentences. The Chinese and English words are listed horizontally and vertically, respectively. The dark points indicate the correspondence between the words in two languages. For example, the first Chinese word "*zhongguo*" is aligned to the fourth English word "*China*".

Formally, given a source sentence $\mathbf{f} = f_1^J = f_1, \ldots, f_j, \ldots, f_J$ and a target sentence $\mathbf{e} = e_1^I = e_1, \ldots, e_i, \ldots, e_I$, we define a link $l = (j, i)$ to exist if f_j and e_i are translation (or part of translation) of one another. Then, an alignment $\mathbf{a}$ is a subset of the Cartesian product of word positions:

$$\mathbf{a} \subseteq \{(j, i) : j = 1, \ldots, J; i = 1, \ldots, I\} \quad (1)$$

Usually, SMT systems only use the 1-best alignments for extracting translation rules. For example, given a source phrase $\tilde{f}$ and a target phrase $\tilde{e}$, the phrase pair $(\tilde{f}, \tilde{e})$ is said to be *consistent* (Och and Ney, 2004) with the alignment if and only if: (1) there must be at least one word inside one phrase aligned to a word inside the other phrase and (2) no words inside one phrase can be aligned to a word outside the other phrase.

After all phrase pairs are extracted from the training corpus, their translation probabilities can be estimated as *relative frequencies* (Och and Ney, 2004):

$$\phi(\tilde{e}|\tilde{f}) = \frac{count(\tilde{f}, \tilde{e})}{\sum_{\tilde{e}'} count(\tilde{f}, \tilde{e}')} \quad (2)$$

where $count(\tilde{f}, \tilde{e})$ indicates how often the phrase pair $(\tilde{f}, \tilde{e})$ occurs in the training corpus.

Besides relative frequencies, *lexical weights* (Koehn et al., 2003) are widely used to estimate how well the words in $\tilde{f}$ translate the words in $\tilde{e}$. To do this, one needs first to estimate a lexical translation probability distribution $w(e|f)$ by relative frequency from the same word alignments in the training corpus:

$$w(e|f) = \frac{count(f, e)}{\sum_{e'} count(f, e')} \quad (3)$$

Note that a special source NULL token is added to each source sentence and aligned to each unaligned target word.

As the alignment $\tilde{a}$ between a phrase pair $(\tilde{f}, \tilde{e})$ is retained during extraction, the lexical weight can be calculated as

$$p_w(\tilde{e}|\tilde{f}, \tilde{a}) = \prod_{i=1}^{|\tilde{e}|} \frac{1}{|\{j|(j, i) \in \tilde{a}\}|} \sum w(e_i|f_j) \quad (4)$$

If there are multiple alignments $\tilde{a}$ for a phrase pair $(\tilde{f}, \tilde{e})$, Koehn et al. (2003) choose the one with the highest lexical weight:

$$p_w(\tilde{e}|\tilde{f}) = \max_{\tilde{a}} \left\{ p_w(\tilde{e}|\tilde{f}, \tilde{a}) \right\} \quad (5)$$

Simple and effective, relative frequencies and lexical weights have become the standard features in modern discriminative SMT systems.

3 Weighted Alignment Matrix

We believe that offering more candidate alignments to extracting translation rules might help improve translation quality. Instead of using n-best lists (Venugopal et al., 2008), we propose a new structure called *weighted alignment matrix*.

We use an example to illustrate our idea. Figure 2(a) and Figure 2(b) show two alignments of a Chinese-English sentence pair. We observe that some links (e.g., (1,4) corresponding to the word

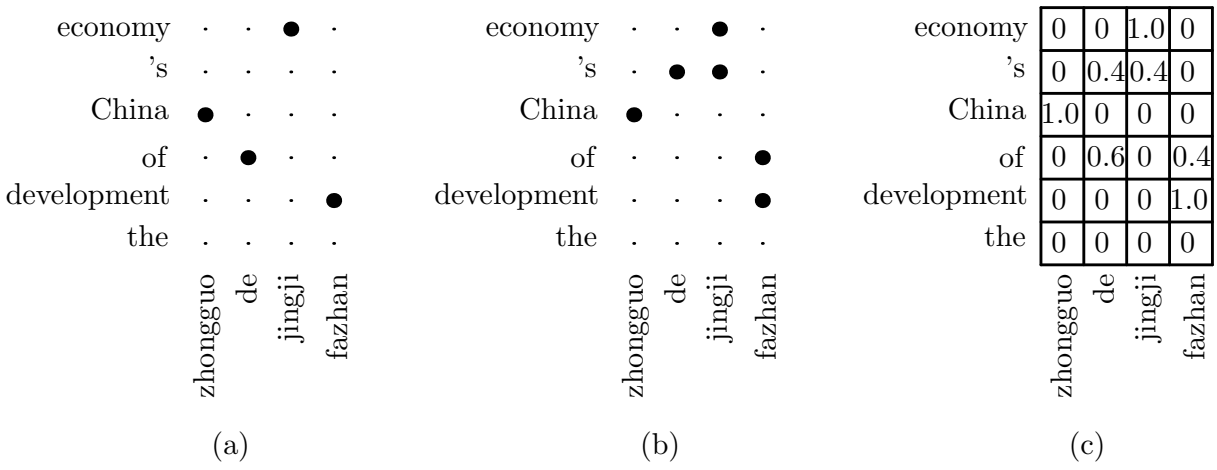


Figure 2: (a) One alignment of a sentence pair; (b) another alignment of the same sentence pair; (c) the resulting weighted alignment matrix that takes the two alignments as samples, of which the initial probabilities are 0.6 and 0.4, respectively.

pair ("*zhongguo*", "*China*")) occur in both alignments, some links (e.g., (2,3) corresponding to the word pair ("*de*","*of*")) occur only in one alignment, and some links (e.g., (1,1) corresponding to the word pair ("*zhongguo*", "*the*")) do not occur. Intuitively, we can estimate how well two words are aligned by calculating its relative frequency, which is the probability sum of alignments in which the link occurs divided by the probability sum of all possible alignments. Suppose that the probabilities of the two alignments in Figures 2(a) and 2(b) are 0.6 and 0.4, respectively. We can estimate the relative frequencies for every word pair and obtain a weighted matrix shown in Figure 2(c). Therefore, each word pair is associated with a probability to indicate how well they are aligned. For example, in Figure 2(c), we say that the word pair ("*zhongguo*", "*China*") is definitely aligned, ("*zhongguo*", "the") is definitely unaligned, and ("*de*", "*of*") has a 60% chance to get aligned.

Formally, a weighted alignment matrix m is a $J \times I$ matrix, in which each element stores a *link probability* $p_m(j,i)$ to indicate how well f_j and e_i are aligned. Currently, we estimate link probabilities from an n-best list by calculating relative frequencies:

$$p_m(j,i) = \frac{\sum_{a \in \mathcal{N}} p(a) \times \delta(a,j,i)}{\sum_{a \in \mathcal{N}} p(a)} \quad (6)$$

$$= \sum_{a \in \mathcal{N}} p(a) \times \delta(a,j,i) \quad (7)$$

where

$$\delta(a,j,i) = \begin{cases} 1 & (j,i) \in a \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

Note that $\mathcal{N}$ is an n-best list, $p(a)$ is the probability of an alignment a in the n-best list, $\delta(a,j,i)$ indicates whether a link (j,i) occurs in the alignment a or not. We assign 0 to any unseen alignment. As $p(a)$ is usually normalized (i.e., $\sum_{a \in \mathcal{N}} p(a) \equiv 1$), we remove the denominator in Eq. (6).

Accordingly, the probability that the two words f_j and e_i are not aligned is

$$\bar{p}_m(j,i) = 1.0 - p_m(j,i) \quad (9)$$

For example, as shown in Figure 2(c), the probability for the two words "*de*" and "*of*" being aligned is 0.6 and the probability that they are not aligned is 0.4.

Intuitively, the probability of an alignment a is the product of link probabilities. If a link (j,i) occurs in a, we use $p_m(j,i)$; otherwise we use $\bar{p}_m(j,i)$. Formally, given a weighted alignment matrix m, the probability of an alignment a can be calculated as

$$p_m(a) = \prod_{j=1}^{J} \prod_{i=1}^{I} (p_m(j,i) \times \delta(a,j,i) + \bar{p}_m(j,i) \times (1 - \delta(a,j,i))) \quad (10)$$

It proves that the sum of all alignment probabilities is always 1: $\sum_{a \in \mathcal{A}} p_m(a) \equiv 1$, where $\mathcal{A}$

```
1: procedure PHRASEEXTRACT(f_1^J, e_1^I, m, l)
2:     R ← ∅
3:     for j_1 ← 1 ... J do
4:         j_2 ← j_1
5:         while j_2 < J ∧ j_2 − j_1 < l do
6:             T ← {i|∃j : j_1 ≤ j ≤ j_2 ∧ p_m(j, i) > 0}
7:             i_l ← MIN(T)
8:             i_u ← MAX(T)
9:             for n ← 1 ... l do
10:                for i_1 ← i_l − n + 1 ... i_u do
11:                    i_2 ← i_1 + n − 1
12:                    R ← R ∪ {(f_{j_1}^{j_2}, e_{i_1}^{i_2})}
13:                end for
14:            end for
15:            j_2 ← j_2 + 1
16:        end while
17:    end for
18:    return R
19: end procedure
```

Figure 3: Algorithm for extracting phrase pairs from a sentence pair $\langle f_1^J, e_1^I \rangle$ annotated with a weighted alignment matrix m.

is the set of all possible alignments. Therefore, a weighted alignment matrix is capable of encoding the probabilities of $2^{J \times I}$ alignments using only a $J \times I$ space.

Note that $p_m(a)$ is not necessarily equal to $p(a)$ because the encoding of a weighted alignment matrix changes the alignment probability distribution. For example, while the initial probability of the alignment in Figure 2(a) (i.e., $p(a)$) is 0.6, the probability of the same alignment encoded in the matrix shown in Figure 2(c) (i.e., $p_m(a)$) becomes 0.1296 according to Eq. (10). It should be emphasized that a weighted matrix encodes all possible alignments rather than the input n-best list, although the link probabilities are estimated from the n-best list.

4 Phrase Pair Extraction

In this section, we describe how to extract phrase pairs from the training corpus annotated with weighted alignment matrices (Section 4.1) and how to estimate their relative frequencies (Section 4.2) and lexical weights (Section 4.3).

4.1 Extraction Algorithm

Och and Ney (2004) describe a "phrase-extract" algorithm for extracting phrase pairs from a sentence pair annotated with a 1-best alignment. Given a source phrase, they first identify the target phrase that is consistent with the alignment. Then, they expand the boundaries of the target phrase if the boundary words are unaligned.

Unfortunately, this algorithm cannot be directly used to manipulate a weighted alignment matrix, which is a compact representation of all possible alignments. The major difference is that the "tight" phrase that has both boundary words aligned is not necessarily the smallest candidate in a weighted matrix. For example, in Figure 2(a), the "tight" target phrase corresponding to the source phrase "*zhongguo de*" is "*of China*". According to Och's algorithm, the target phrase "*China*" breaks the alignment consistency and therefore is not valid candidate. However, this is not true for using the weighted matrix shown in Figure 2(c). The target phrase "*China*" is treated as a "potential" candidate [1], although it might be assigned only a small fractional count (see Table 1).

Therefore, we enumerate all potential phrase pairs and calculate their fractional counts for eliminating less promising candidates. Figure 3 shows the algorithm for extracting phrases from a weighted matrix. The input of the algorithm is a source sentence f_1^J, a target sentence e_1^I, a weighted alignment matrix m, and a phrase length limit l (line 1). After initializing $\mathcal{R}$ that stores collected phrase pairs (line 2), we identify the corresponding target phrases for all possible source phrases (lines 3-5). Given a source phrase $f_{j_1}^{j_2}$, we find the lower and upper bounds of target positions (i.e., i_l and i_u) that have positive link probabilities (lines 6-8). For example, the lower bound is 3 and the upper bound is 5 for the source phrase "*zhongguo de*" in Figure 2(c). Finally, we enumerate all target phrases that allow for unaligned boundary words with varying phrase lengths (lines 9-14). Note that we need to ensure that $1 \leq i_1 \leq I$ and $1 \leq i_2 \leq I$ in lines 10-11, which are omitted for simplicity.

4.2 Calculating Relative Frequencies

To estimate the relative frequency of a phrase pair, we need to estimate how often it occurs in the training corpus. Given an n-best list, the fractional count of a phrase pair is the probability sum of the alignments with which the phrase pair is consistent. Obviously, it is unrealistic for a weighted alignment matrix to enumerate all possible alignments explicitly to calculate fractional counts. Instead, we resort to link probabilities to calculate

[1] By potential, we mean that the fractional count of a phrase pair is positive. Section 4.2 describes how to calculate fractional counts.

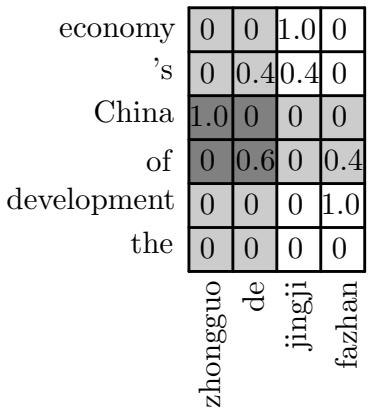


Figure 4: An example of calculating fractional count. Given the phrase pair ("*zhongguo de*", "*of China*"), we divide the matrix into three areas: inside (heavy shading), outside (light shading), and irrelevant (no shading).

counts efficiently. Equivalent to explicit enumeration, we interpret the fractional count of a phrase pair as the probability that it satisfies the two alignment consistency conditions (see Section 2).

Given a phrase pair, we divide the elements of a weighted alignment matrix into three categories: (1) *inside* elements that fall inside the phrase pair, (2) *outside* elements that fall outside the phrase pair while fall in the same row or the same column, and (3) *irrelevant* elements that fall outside the phrase pair while fall in neither the same row nor the same column. Figure 4 shows an example. Given the phrase pair ("*zhongguo de*", "*of China*"), we divide the matrix into three areas: inside (heavy shading), outside (light shading), and irrelevant (no shading).

To what extent a phrase pair satisfies the alignment consistency is measured by calculating *inside* and *outside* probabilities. Although there are the same terms in the parsing literature, they have different meanings here. The inside probability indicates the chance that there is at least one word inside one phrase aligned to a word inside the other phrase. The outside probability indicates the chance that no words inside one phrase are aligned to a word outside the other phrase.

Given a phrase pair $(f_{j_1}^{j_2}, e_{i_1}^{i_2})$, we denote the inside area as $in(j_1, j_2, i_1, i_2)$ and the outside area as $out(j_1, j_2, i_1, i_2)$. Therefore, the inside probability of a phrase pair is calculated as

$$\alpha(j_1, j_2, i_1, i_2) = 1 - \prod_{(j,i) \in in(j_1, j_2, i_1, i_2)} \bar{p}_m(j, i) \quad (11)$$

target phrase	α	β	$count$
of China	1.0	0.36	0.36
of China 's	1.0	0.36	0.36
China 's	1.0	0.24	0.24
China	1.0	0.24	0.24
's economy	0.4	0	0

Table 1: Some candidate target phrases of the source phrase "*zhongguo de*" in Figure 4, where α is inside probability, β is outside probability, and $count$ is fractional count.

For example, the inside probability for ("*zhongguo de*", "*of China*") in Figure 4 is 1.0, which means that there always exists at least one aligned word pair inside.

Accordingly, the outside probability of a phrase pair is calculated as

$$\beta(j_1, j_2, i_1, i_2) = \prod_{(j,i) \in out(j_1, j_2, i_1, i_2)} \bar{p}_m(j, i) \quad (12)$$

For example, the outside probability for ("*zhongguo de*", "*of China*") in Figure 4 is 0.36, which means the probability that there are no aligned word pairs outside is 0.36.

Finally, we use the product of inside and outside probabilities as the fractional count of a phrase pair:

$$count(f_{j_1}^{j_2}, e_{i_1}^{i_2}) = \alpha(j_1, j_2, i_1, i_2) \times \beta(j_1, j_2, i_1, i_2) \quad (13)$$

Table 1 lists some candidate target phrases of the source phrase "*zhongguo de*" in Figure 4. We also give their inside probabilities, outside probabilities, and fractional counts.

After collecting the fractional counts from the training corpus, we then use Eq. (2) to calculate relative frequencies in two translation directions.

Often, our approach extracts a large amount of phrase pairs from training corpus as we soften the alignment consistency constraint. To maintain a reasonable phrase table size, we discard any phrase pair that has a fractional count lower than a threshold t. During extraction, we first obtain a list of candidate target phrases for each source phrase, as shown in Table 1. Then, we prune the list according to the threshold t. For example, we only retain the top two candidates in Table 1 if $t = 0.3$. Note that we perform the pruning locally. Although it is more reasonable to prune a phrase table after accumulating all fractional counts from

training corpus, such global pruning strategy usually leads to very large disk and memory requirements.

4.3 Calculating Lexical Weights

Recall that we need to obtain two translation probability tables $w(e|f)$ and $w(f|e)$ before calculating lexical weights (see Section 2). Following Koehn et al. (2003), we estimate the two distributions by relative frequencies from the training corpus annotated with weighted alignment matrices. In other words, we still use Eq. (3) but the way of calculating fractional counts is different now.

Given a source word f_j, a target word e_i, and a weighted alignment matrix, the fractional count $count(f_j, e_i)$ is $p_m(j, i)$. For NULL words, the fractional counts can be calculated as

$$count(f_j, e_0) = \prod_{i=1}^{I} \bar{p}_m(j, i) \quad (14)$$

$$count(f_0, e_i) = \prod_{j=1}^{J} \bar{p}_m(j, i) \quad (15)$$

For example, in Figure 4, $count$(*de*, *of*) is 0.6, $count$(*de*,*NULL*) is 0.24, and $count$(*NULL*,*of*) is 0.24.

Then, we adapt Eq. (4) to calculate lexical weight:

$$p_w(\tilde{e}|\tilde{f}, m) = \prod_{i=1}^{|\tilde{e}|} \left(\left(\frac{1}{\{j|p_m(j,i) > 0\}} \times \sum_{\forall j: p_m(j,i)>0} p(e_i|f_j) \times p_m(j, i) \right) + p(e_i|f_0) \times \prod_{j=1}^{|\tilde{f}|} \bar{p}_m(j, i) \right) \quad (16)$$

For example, for the target word "*of*" in Figure 4, the sum of aligned and unaligned probabilities is

$$\frac{1}{2} \times (p(\textit{of}|\textit{de}) \times 0.6 + p(\textit{of}|\textit{fazhan}) \times 0.4) + p(\textit{of}|\text{NULL}) \times 0.24$$

Note that we take link probabilities into account and calculate the probability that a target word translates a source NULL token explicitly.

5 Experiments

5.1 Data Preparation

We evaluated our approach on Chinese-to-English translation. We used the FBIS corpus (6.9M + 8.9M words) as the training data. For language model, we used the SRI Language Modeling Toolkit (Stolcke, 2002) to train a 4-gram model on the Xinhua portion of GIGAWORD corpus. We used the NIST 2002 MT evaluation test set as our development set, and used the NIST 2005 test set as our test set. We evaluated the translation quality using *case-insensitive* BLEU metric (Papineni et al., 2002).

To obtain weighted alignment matrices, we followed Venugopal et al. (2008) to produce n-best lists via GIZA++. We first ran GIZA++ to produce 50-best lists in two translation directions. Then, we used the refinement technique "grow-diag-final-and" (Koehn et al., 2003) to all 50×50 bidirectional alignment pairs. Suppose that p_{s2t} and p_{t2s} are the probabilities of an alignment pair assigned by GIZA++, respectively. We used $p_{s2t} \times p_{t2s}$ as the probability of the resulting symmetric alignment. As different alignment pairs might produce the same symmetric alignments, we followed Venugopal et al. (2008) to remove duplicate alignments and retain only the alignment with the highest probability. Therefore, there were 550 candidate alignments on average for each sentence pair in the training data. We obtained n-best lists by selecting the top n alignments from the 550-best lists. The probability of each alignment in the n-best list was re-estimated by re-normalization (Venugopal et al., 2008). Finally, these n-best alignments served as samples for constructing weighted alignment matrices.

After extracting phrase pairs from n-best lists and weighted alignment matrices, we ran Moses (Koehn et al., 2007) to translate the development and test sets. We used the simple distance-based reordering model to remove the dependency of lexicalization on word alignments for Moses.

5.2 Effect of Pruning Threshold

Our first experiment investigated the effect of pruning threshold on translation quality (BLEU scores on the test set) and the phrase table size (filtered for the test set), as shown in Figure 5. To save time, we extracted phrase pairs just from the first 10K sentence pairs of the FBIS corpus. We used 12 different thresholds: 0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9. Obviously, the lower the threshold is, the more phrase pairs are extracted. When $t = 0.0001$, the number of phrase pairs used on the test set was 460,284

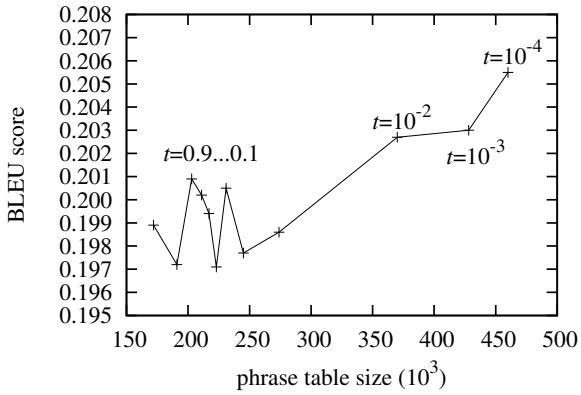


Figure 5: Effect of pruning threshold on translation quality and phrase table size.

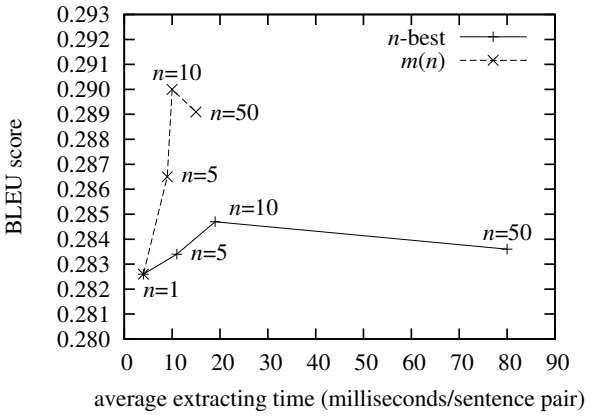


Figure 6: Comparison of n-best alignments and weighted alignment matrices. We use $m(n)$ to denote the matrices that take n-best lists as samples.

and the BLEU score was 20.55. Generally, both the number of phrase pairs and the BLEU score went down with the increase of t. However, this trend did not hold within the range [0.1, 0.9]. To achieve a good tradeoff between translation quality and phrase table size, we set $t = 0.01$ for the following experiments.

5.3 N-best lists Vs. Weighted Matrices

Figure 6 shows the BLEU scores and average extraction time using n-best alignments and weighted matrices, respectively. We used the entire training data for phrase extraction. When using 1-best alignments, Moses achieved a BLEU score of 0.2826 and the average extraction time was 4.19 milliseconds per sentence pair (see point $n = 1$). The BLEU scores rose with the increase of n for using n-best alignments. However, the score went down slightly when $n = 50$. This suggests that including more noisy alignments might be harmful. These improvements over 1-best alignments are not statistically significant. This finding failed to echo the promising results reported by Venogopal et al. (2008). We think that there are two possible reasons. First, they evaluated their approach on the IWSLT data while we used the NIST data. It might be easier to obtain significant improvements on the IWSLT data in which the sentences are shorter. Second, they used the hierarchical phrase-based system while we used the phrase-based system, which might be less sensitive to word alignments because the alignments inside the phrase pairs hardly have an effect.

When using weighted alignment matrices, we obtained higher BLEU scores than using n-best lists with much less extraction time. We achieved a BLEU score of 0.2901 when using the weighted matrices estimated from 10-best lists. The absolute improvement of 0.75 over using 1-best alignments (from 0.2826 to 0.2901) is statistically significant at $p < 0.05$ by using *sign-test* (Collins et al., 2005). Although the improvements over n-best lists are not always statistically significant, weighted alignment matrices maintain consistent superiority in both translation quality and extraction speed.

5.4 Comparison of Parameter Estimation

In theory, the set of phrase pairs extracted from n-best alignments is the subset of the set extracted from the corresponding weighted matrices. In practice, however, this is not true because we use the pruning threshold t to maintain a reasonable table size. Even so, the phrase tables produced by n-best lists and weighted matrices still share many phrase pairs.

Table 2 gives some statistics. We use m(10) to represent the weighted matrices estimated from 10-best lists. "all" denotes the full phrase table, "shared" denotes the intersection of two tables, and "non-shared" denotes the complement. Note that the probabilities of "shared" phrase pairs are different for the two approaches. We obtained 6.13M and 6.34M phrase pairs for the test set by using 10-best lists and the corresponding matrices, respectively. There were 4.58M phrase pairs included by both tables. Note that the relative frequencies and lexical weights for the same phrase

method	shared		non-shared		all	
	phrases	BLEU	phrases	BLEU	phrases	BLEU
10-best	4.58M	**28.35**	1.55M	12.32	6.13M	28.47
m(10)	4.58M	**28.90**	1.76M	13.21	6.34M	29.01

Table 2: Comparison of phrase tables learned from n-best lists and weighted matrices. We use m(10) to represent the weighted matrices estimated from 10-best lists. "all" denotes the full phrase table, "shared" denotes the intersection of two tables, and "non-shared" denotes the complement. Note that the probabilities of "shared" phrase pairs are different for the two approaches.

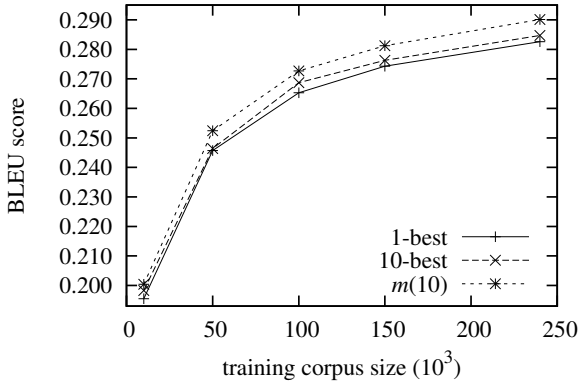


Figure 7: Comparison of n-best alignments and weighted alignment matrices with varying training corpus sizes.

pairs might be different in two tables. We found that using matrices outperformed using n-best lists even with the same phrase pairs. This suggests that our methods for parameter estimation make better use of noisy data. Another interesting finding was that using the shared phrase pairs achieved almost the same results with using full phrase tables.

5.5 Effect of Training Corpus Size

To investigate the effect of training corpus size on our approach, we extracted phrase pairs from n-best lists and weighted matrices trained on five training corpora with varying sizes: 10K, 50K, 100K, 150K, and 239K sentence pairs. As shown in Figure 7, our approach outperformed both 1-best and n-best lists consistently. More importantly, the gains seem increase when more training data are used.

5.6 Results on Other Language Pairs

To further examine the efficacy of the proposed approach, we scaled our experiments to large data with multiple language pairs. We used the Europarl training corpus from the WMT07 shared task. [2] Table 3 shows the statistics of the training data. There are four languages (Spanish, French, German, and English) and six translation directions (Foreign-to-English and English-to-Foreign). We used the "dev2006" data in the "dev" directory as the development set and the "test2006" data in the "devtest" directory as the test set. Both the development and test sets contain 2,000 sentences with single reference translations.

We tokenized and lowercased all the training, development, and test data. We trained a 4-gram language model using SRI Language Modeling Toolkit on the target side of the training corpus for each task. We ran GIZA++ on the entire training data to obtain n-best alignments and weighted matrices. To save time, we just used the first 100K sentences of each aligned training corpus to extract phrase pairs.

	S↔E	F↔E	G↔E
Sentences	1.26M	1.29M	1.26M
Foreign words	33.16M	33.18M	29.58M
English words	31.81M	32.62M	31.93M

Table 3: Statistics of the Europarl training data. "S" denotes Spanish, "E" denotes English, "F" denotes French, "G" denotes German.

	1-best	10-best	m(10)
S→E	30.90	30.97	31.03
E→S	31.16	31.25	31.34
F→E	30.69	30.76	30.82
E→F	26.42	26.65	26.54
G→E	24.46	24.58	24.66
E→G	18.03	18.30	18.20

Table 4: BLEU scores (case-insensitive) on the Europarl data. "S" denotes Spanish, "E" denotes English, "F" denotes French, "G" denotes German.

[2] http://www.statmt.org/wmt07/shared-task.html

Table 4 lists the case-insensitive BLEU scores of 1-best, 10-best, and m(10) on the Europarl data. Using weighted packed matrices continued to show advantage over using 1-best alignments on multiple language pairs. However, these improvements were very small and not significant. We attribute this to the fact that GIZA++ usually produces high quality 1-best alignments for closely-related European language pairs, especially when trained on millions of sentences.

6 Related Work

Recent studies has shown that SMT systems can benefit from making the annotation pipeline wider: using packed forests instead of 1-best trees (Mi et al., 2008), word lattices instead of 1-best segmentations (Dyer et al., 2008), and n-best alignments instead of 1-best alignments (Venugopal et al., 2008). We propose a compact representation of multiple word alignments that enables SMT systems to make a better use of noisy alignments.

Matusov et al. (2004) propose "cost matrices" for producing symmetric alignments. Kumar et al. (2007) describe how to use "posterior probability matrices" to improve alignment accuracy via a bridge language. Although not using the term "weighted matrices" directly, they both assign a probability to each word pair.

We follow Och and Ney (2004) to develop a new phrase extraction algorithm for weighted alignment matrices. The methods for calculating relative frequencies (Och and Ney, 2004) and lexical weights (Koehn et al., 2003) are also adapted for the weighted matrix case.

Many researchers (e.g., (Venugopal et al., 2003; Deng et al., 2008)) observe that softening the alignment consistency constraint help improve translation quality. For example, Deng et al. (2008) define a feature named "within phrase pair consistency ratio" to measure the degree of consistency. As each link is associated with a probability in a weighted matrix, we use these probabilities to evaluate the validity of a phrase pair.

We estimate the link probabilities by calculating relative frequencies over n-best lists. Niehues and Vogel (2008) propose a discriminative approach to modeling the alignment matrix directly. The difference is that they assign a boolean value instead of a probability to each word pair.

7 Conclusion and Future Work

We have presented a new structure called weighted alignment matrix that encodes the alignment distribution for a sentence pair. Accordingly, we develop new methods for extracting phrase pairs and estimating their probabilities. Our experiments show that the proposed approach achieves better translation quality over using n-best lists in less extraction time. An interesting finding is that our approach performs better than the baseline even they use the same phrase pairs.

Although our approach consistently outperforms using 1-best alignments for varying language pairs, the improvements are comparatively small. One possible reason is that taking n-best lists as samples sometimes might change alignment probability distributions inappropriately. A more principled solution is to directly model the weighted alignment matrices, either in a generative or a discriminative way. We believe that better estimation of alignment distributions will result in more significant improvements.

Another interesting direction is applying our approach to extracting translation rules with hierarchical structures such as hierarchical phrases (Chiang, 2007) and tree-to-string rules (Galley et al., 2006; Liu et al., 2006). We expect that these syntax-based systems could benefit more from our approach.

Acknowledgement

The authors were supported by Microsoft Research Asia Natural Language Processing Theme Program grant (2009-2010), High-Technology R&D Program (863) Project No. 2006AA010108, and National Natural Science Foundation of China Contract 60736014. Part of this work was done while Yang Liu was visiting the SMT group led by Stephan Vogel at CMU. We thank the anonymous reviewers for their insightful comments. We are also grateful to Stephan Vogel, Alon Lavie, Francisco Guzman, Nguyen Bach, Andreas Zollmann, Vamshi Ambati, and Kevin Gimpel for their helpful feedback.

References

Phil Blunsom and Trevor Cohn. 2006. Discriminative word alignment with conditional random fields. In *Proceedings of COLING/ACL 2006*, pages 65–72, Sydney, Australia, July.

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

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL 2005*, pages 531–540, Ann Arbor, USA, June.

Yonggang Deng, Jia Xu, and Yuqing Gao. 2008. Phrase table training for precision and recall: What makes a good phrase and a good phrase pair? In *Proceedings of ACL/HLT 2008*, pages 81–88, Columbus, Ohio, USA, June.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL/HLT 2008*, pages 1012–1020, Columbus, Ohio, June.

Alexander Fraser and Daniel Marcu. 2007. Measuring word alignment quality for statistical machine translation. *Computational Linguistics, Squibs and Discussions*, 33(3):293–303.

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

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL 2003*, pages 127–133, Edmonton, Canada, May.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL 2007 (poster)*, pages 77–80, Prague, Czech Republic, June.

Shankar Kumar, Franz J. Och, and Wolfgang Macherey. 2007. Improving word alignment with bridge languages. In *Proceedings of EMNLP 2007*, pages 42–50, Prague, Czech Republic, June.

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of ACL 2005*, pages 459–466, Ann Arbor, Michigan, June.

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

Evgeny Matusov, Richard Zens, and Hermann Ney. 2004. Symmetric word alignments for statistical machine translation. In *Proceedings of COLING 2004*, pages 219–225, Geneva, Switzerland, August.

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

Robert C. Moore, Wen-tau Yih, and Andreas Bode. 2006. Improved discriminative bilingual word alignment. In *Proceedings of COLING/ACL 2006*, pages 513–520, Sydney, Australia, July.

Jan Niehues and Stephan Vogel. 2008. Discriminative word alignment via alignment matrix modeling. In *Proceedings of WMT-3*, pages 18–25, Columbus, Ohio, USA, June.

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

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

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

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL/HLT 2008*, pages 577–585, Columbus, Ohio, June.

Andreas Stolcke. 2002. Srilm - an extension language model modeling toolkit. In *Proceedings of ICSLP 2002*, pages 901–904, Denver, Colorado, September.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 2005. A discriminative matching approach to word alignment. In *Proceedings of HLT/EMNLP 2005*, pages 73–80, Vancouver, British Columbia, Canada, October.

Ashish Venugopal, Stephan Vogel, and Alex Waibel. 2003. Effective phrase translation extraction from alignment models. In *Proceedings of ACL 2003*, pages 319–326, Sapporo, Japan, July.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. 2008. Wider pipelines: n-best alignments and parses in mt training. In *Proceedings of AMTA 2008*, pages 192–201, Waikiki, Hawaii, October.

Stephan Vogel and Hermann Ney. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of COLING 1996*, pages 836–841, Copenhagen, Danmark, August.

Sinuhe — Statistical Machine Translation using a Globally Trained Conditional Exponential Family Translation Model

Matti Kääriäinen
Department of Computer Science
FI-00014 University of Helsinki, Finland
matti.kaariainen@cs.helsinki.fi

Abstract

We present a new phrase-based conditional exponential family translation model for statistical machine translation. The model operates on a feature representation in which sentence level translations are represented by enumerating all the known phrase level translations that occur inside them. This makes the model a good match with the commonly used phrase extraction heuristics. The model's predictions are properly normalized probabilities. In addition, the model automatically takes into account information provided by phrase overlaps, and does not suffer from reference translation reachability problems.

We have implemented an open source translation system Sinuhe based on the proposed translation model. Our experiments on Europarl and GigaFrEn corpora demonstrate that finding the unique MAP parameters for the model on large scale data is feasible with simple stochastic gradient methods. Sinuhe is fast and memory efficient, and the BLEU scores obtained by it are only slightly inferior to those of Moses.

1 Introduction

In current phrase-based statistical machine translation systems such as Moses[1] (Koehn et al., 2007), the translation model is defined in terms of phrase pairs (biphrases) extracted from a bilingual corpus as follows. The corpus is first word-aligned using a word alignment heuristic (Och and Ney, 2003). The phrase extraction heuristic then extracts all the biphrases that are compatible with the word alignment (Och et al., 1999). This way, each sentence pair may generate any number of potentially overlapping biphrases. However, when defining the phrase-based sentence level translation model, phrase overlaps are explicitly disallowed: The source sentence is segmented into disjoint phrases, which are translated independently using conditional phrase-level translation models that have been estimated from extracted biphrase counts.

The disparity between the phrase extraction heuristic and the use of the extracted biphrases can be addressed in at least three ways. One approach is to simply ignore the disparity as is done, e.g., in Moses. While empirically succesful, this approach is hard to justify theoretically, and begs the question of whether more principled methods might lead to better translation results. The other extensively studied approach is to replace the phrase extraction heuristic with a method that better matches the use of the extracted phrases (see, e.g., (Marcu and Wong, 2002; DeNero et al., 2008) and the references therein). While theoretically sound, this approach is computationally challenging both in practice (DeNero et al., 2008) and in theory (DeNero and Klein, 2008), may suffer from reference reachability problems (DeNero et al., 2006), and in the end may lead to inferior translation quality (Koehn et al., 2003).

In this paper, we study a third alternative. We propose a new translation model that is compatible with the phrase extraction heuristic. The proposed machine learning inspired translation model takes the form of a conditional exponential family probability distribution over a feature representation for word-aligned sentence pairs. The feature representation represents a word-aligned sentence pair by essentially enumerating the (multi)set of biphrases that would have been extracted from it,

[1] Throughout this paper, we refer to Moses for concreteness, but most of the discussion applies to other standard phrase-based statistical machine translation systems as well.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1027–1036,
Singapore, 6-7 August 2009.

together with the source positions at which the biphrases occur. The model's predictions are conditional probabilities for such sets of biphrases given the source sentence.

The chosen feature representation has many advantages. Since all word-aligned sentence pairs can be represented, reference reachability problems are automatically circumvented. For example, if the translation of a sentence consisted solely of words that do not occur in the phrase table, then the feature vector for the translation would be the all zero vector. As the training data receives nonzero probability, maximum likelihood or maximum a posteriori (MAP) parameters for the model can be estimated in a principled way without resorting to pseudo-references. The fact that the model is not restricted to using disjoint biphrases means that the information in biphrase overlaps is automatically taken into account. This may help in smoothing the model's predictions on long and rare phrases, and in enhancing fluency at places that otherwise would be phrase boundaries. Also, the model can be extended in a principled way by introducing additional features (e.g., translations from a dictionary, biphrases with gaps, biphrases over POS tags,...).

The proposed model has one parameter per biphrase feature, so the total number of parameters is easily millions or more. Still, the model structure is designed so that feature expectations and related quantities can be computed efficiently by dynamic programming. It is thus feasible to compute the gradient of the MAP objective, and simple gradient ascent can be used to efficiently find the globally optimal model parameters (with respect to a suitably scaled Gaussian prior used for regularization). Exact inference is also possible by dynamic programming when translations are predicted by the translation model alone. When other features like a language model are included, one needs to resort to beam search type approximate dynamic programming for decoding.

We have implemented a translation system called `Sinuhe` based on the proposed translation model. The system has been released under the GPLv3 open source license (Kääriäinen, 2009). Our experiments on Europarl and GigaFrEn corpora demonstrate that the proposed translation model scales well to large data, and offers translation quality that is only slightly worse than that of the baseline system `Moses`. In terms of translation speed, `Sinuhe` is already clearly better.

The rest of this paper is organized as follows. After briefly reviewing related work in Section 2, we describe the proposed translation model in Section 3. Finally, experimental results are presented in Section 4, and conclusions in Section 5.

2 Related work

The proposed translation model is strongly influenced by machine learning techniques for solving sequence prediction tasks, most notably the work on conditional random fields (Lafferty et al., 2001). The modelling task in machine translation is, however, more complicated than sequence labelling (not one-to-one, reorderings), so the standard methods cannot be directly applied here. The model we propose is also related to standard phrase-based translation models through the use of the same phrase-level translation features. However, the way we use the features is quite different.

There exists a number of discriminative approaches whose model structure, training criteria, or both, are similar to ours. However, to our knowledge, none of the other systems operates directly on biphrase features, scales up to bilingual corpora with millions of sentence pairs, and achieves translation quality comparable to fully tuned standard phrase-based systems. The approach most closely resembling ours is the independently developed global discriminative log-linear model based on synchronous context-free grammars (Blunsom and Osborne, 2008; Blunsom et al., 2008). The version presented in (Blunsom and Osborne, 2008) operates on millions of rule count features analogous to our biphrase features, and integrates a language model into training and decoding. The system can be trained on tens of thousands of short sentences yielding better translations than a baseline system `Hiero` on this data. The version presented in (Blunsom et al., 2008) scales to more than a hundred thousand short training sentences, but does not integrate a language model and thus has performance that improves upon `Hiero` without a language model only. Both versions deal with derivational ambiguity by treating derivations as a latent variables that are integrated out to get conditional probabilities for translations[2]. The downside of this

[2]In our translation model, coping with multiple derivations is not needed as there is just one derivation for each word-aligned sentence pair. However, dealing with alternative word alignments might be beneficial, though as argued

is that approximations are needed in computing the maximum probability translation in decoding, and also in computing model expectations in training when a language model is used. In addition, since the models operate directly on translations, using probabilistic training criteria for learning the model parameters is possible only if all reference translations in the training data can be generated by the model. In practice, this problem can be circumvented by discarding the training sentence pairs with unreachable reference translations, but this may mean a significant reduction in the amount of training data (24% in (Blunsom et al., 2008)).

Another closely related approach is the independently developed discriminative block bigram prediction model presented in (Tillmann and Zhang, 2007). This work proposes a global phrase-based translation model very similar to ours, but due to computational reasons, resorts to a localized approximation thereof, and is restricted to biphrases of length at most two. In (Liang et al., 2006) a standard phrase-based model is augmented with more than a million features whose weights are trained discriminatively by a variant of the perceptron algorithm. Reference reachability is again a problem, and the method has not been scaled up to use biphrase features directly.

3 The proposed translation model

3.1 Biphrase extraction

The biphrases used in `Sinuhe` are extracted from the training data with the `Moses` phrase extraction heuristics. The sentence-aligned training corpus S is first word-aligned by running Giza++ in both directions and then symmetrizing the alignments. This maps the original aligned sentence pairs (x, y) into word-aligned sentence pairs (x, a, y), where a is a many-to-many alignment between the words in x and y. Second, using a heuristic proposed in (Och et al., 1999), all the aligned phrase pairs (x', a', y') satisfying the following criteria are extracted: (1) x' and y' consist of consecutive words of x and y, and both have length at most k, (2) a' is the alignment between words of x' and y' induced by a, (3) a' contains at least one link, and (4) there are no links in a that have just one end in x' or y'. Each aligned training sentence (x, a, y) thus generates a number of potentially overlapping aligned biphrase features (x', a', y'). In our experiments, we chose $k = 7$ which is the default in `Moses`. Unlike in `Moses`, we do not map the aligned biphrases (x', a', y') back to non-aligned biphrases (x', y').

To reduce the number of extracted biphrases, for each source phrase x', only biphrases (x', a', y') whose occurrence count is among the top $K = 20$ in the training data are retained (rank ties broken by including all biphrases with rank equal to the limit K). For technical reasons related to our dynamic programming algorithms, we also drop biphrases whose source phrase begins or ends with unlinked words. Finally, we drop all biphrases that occur only once in the training data. This can be motivated by a leave-one-out argument (cf the derivation of Good-Turing estimates): Dropping the biphrases that occur only once in the training data means that the feature representation for a training sentence pair (see Section 3.2) contains only biphrases that occur also in other training examples. Without the leave-one-out pruning, the feature vectors for training sentence pairs would be maximally dense, whereas such feature density cannot be expected on test data. Our system can also be used without the leave-one-out pruning, but according to our preliminary experiments this has little effect on translation quality. An exception seems to be morphologically rich languages with scarce training data on which pruning seems to reduce translation quality.

All the pruning steps combined reduce the phrase table size considerably, but in our experiments, millions of biphrases per language pair still remain (2-4 million for Europarl data and over 95 million for GigaFrEn data).

3.2 Features

Our primary feature representation is a binary feature vector that indicates which aligned biphrases in the phrase table occur in an aligned sentence pair and where. More specifically, a source sentence x aligned to a target sentence y by an alignment a is represented by a binary feature vector $\phi(x, a, y)$ whose component $\phi(x, a, y)_{(x', a', y'), i}$ is 1 iff the aligned biphrase (x', a', y') occurs at source position i in (x, a, y), and 0 otherwise. Here, (x', a', y') occurs in (x, a, y) at source position i iff the phrase extraction process described in Section 3.1 would have extracted it from (x, a, y) at source position i.

in (DeNero et al., 2006), the ambiguity in word alignment is less prevalent than in phrase segmentation.

The weights for the aligned biphrases are tied together by mapping the binary feature vector ϕ (indexed by pairs of an aligned biphrase and a source position) to an integral feature vector $\tilde{\phi}$ (indexed by aligned biphrases only) using the formula $\tilde{\phi}_{(x',a',y')} = \sum_i \phi_{(x',a',y'),i}$. The "real" features that drive the translation process are thus the lowest level binary features ϕ, whereas the higher level representation $\tilde{\phi}$ is convenient in defining the conditional probabilities given by the translation model.

3.3 The model

Instead of modelling the conditional distribution $P(y|x)$ directly, we model the conditional distribution $P(\phi(x,a,y)|x)$ by the following conditional exponential model:

$$P(\phi(x,a,y)|x) = \frac{\exp(w \cdot \tilde{\phi}(x,a,y))}{\sum_{\phi \in \Phi_x} \exp(w \cdot \tilde{\phi})}.$$

Here, w is a parameter vector with one component for each aligned biphrase feature in the phrase table. The set Φ_x defines the set of possible predictions given x, and includes all feature vectors ϕ satisfying the following criteria:

1. There exists a translation y' and an alignment a' such that all active features in ϕ occur in (x, a', y')
2. Features corresponding to aligned biphrases that occur inside aligned biphrases whose features are active in ϕ are also active in ϕ.

Thus, the set Φ_x has a feature representation for all possible aligned sentence pairs (x, a', y') that have x as the source side, so all reference translations y' word-aligned to x in any way a' are representable by features in Φ_x. By condition 1, the predictions given by the model never contain conflicting biphrases, so given any prediction of the model, there always exists a translation y' where all the predicted biphrases do occur. However, since our dynamic programming algorithms can only force active super-phrases implying active sub-phrases (condition 2) but not active sub-phrases implying active super-phrases, the set Φ_x also contains some feature vectors in which the latter type of implications are not enforced. Having such redundant representations for some translations is a waste of probability mass, but we hope it has little effect in practice.

The choice of modelling $P(\phi(x,a,y)|x)$ instead of modelling $P(y|x)$ directly is crucial, both from a modelling and from a computational perspective. From the modelling perspective, the crucial point is that in our approach, any aligned sentence pair (x,a,y) has an associated feature vector $\phi(x,a,y) \in \Phi_x$ that is reachable (i.e., receives non-zero probability) by the model. This means it is straightforward to use probabilistic criteria in learning the model parameters. In contrast, systems modelling $P(y|x)$ directly are often plagued by the reference reachability problem. To use probabilistic training criteria for such systems one needs to circumvent the reference reachability problem, e.g., by using pseudo-references or by dropping out the non-reachable portion of training data.

Working with the feature vectors $\phi(x,a,y)$ instead of working with a and y directly means that we model the ordering and choice of words in y only partially. This way, when computing the normalizing constants and feature expectations, we can partition the unbounded set of potential translations y and alignments a into a smaller set of equivalence classes given by $\phi(x,a,y)$. Though the number of feature vectors $\phi \in \Phi_x$ may be large (exponential in length of x), all the necessary computations can be done exactly and efficiently by dynamic programming. For more details, see Section 3.4.3.

3.4 Learning the model parameters

3.4.1 The objective

We use maximum a posteriori (MAP) estimation to estimate the model parameters w. To control overfitting, we regularize the parameters by a suitably scaled Gaussian prior. This can be also viewed as $L2$ regularization. The prior guarantees that the MAP parameters are unique, and models our belief that the observed feature occurrence counts randomly deviate from their "true" values roughly proportionally to the standard deviations of the occurrence count distributions. The prior variance $\sigma^2_{(x',a',y')}$ for feature (x',a',y') is given by the formula $\sigma^2_{(x',a',y')} = \alpha/\rho_{(x',a',y')}$, where $\alpha > 0$ is a free regularization parameter, and $\rho^2_{(x',a',y')}$ is an empirical estimate of the variance of the occurrence count of (x',a',y') in the training data. This is similar to (Chen and Rosenfeld, 2000), except that we use standard deviations in place of variances. As the estimate for the vari-

ance of a feature we use the occurrence count of the corresponding biphrase in the training data. This could be justified by assuming that the occurrence counts follow a Poisson distribution. We have also run preliminary experiments with other forms of regularization (different ways of computing $\sigma_{(x',a',y')}$, exponential priors corresponding to $L1$ regularization, no regularization), and it looks like the system is not very sensitive to the chosen prior.

Combining the prior with the model, we see that the negative log-posterior $\mathcal{L}(w)$ is given by the formula

$$\sum_{(x',a',y')} \frac{w^2_{(x',a',y')}}{2\sigma^2_{(x',a',y')}} - \sum_{(x,a,y)\in S} \log P(\phi(x,a,y)|x) + C,$$

where the sum over (x', a', y') is understood to go over all aligned biphrase features in the model. This is our criterion for learning w.

3.4.2 Optimization

We solve the optimization problem related to learning w by first order gradient ascent methods. The gradient $\nabla\mathcal{L}(w)$ of $\mathcal{L}(w)$ with respect to w can be written as

$$\sum_{(x',a',y')} \frac{w_{(x',a',y')}}{\sigma^2_{(x',a',y')}} - \sum_{(x,a,y)\in S} \left[\tilde{\phi}(x,a,y) - \mathbb{E}_w[\tilde{\phi}|x]\right],$$

where $\mathbb{E}_w[\tilde{\phi}|x]$ denotes the conditional expectation of the aligned biphrase occurrence count features given x with respect to model parameters w. Feature expectations can be computed by combining the results of a left-to-right and right-to-left dynamic programming sweep over the source sentence. For more details, see Section 3.4.3.

Inspired by the empirical results in (Vishwanathan et al., 2006), we use classic stochastic gradient ascent to solve the optimization problem. At each step t, we sample with replacement a batch S_t of b examples from S. We start from $w_0 = 0$, and use the update rule

$$w_{t+1} = w_t - \eta_t \nabla\mathcal{L}_t(w_t), \qquad (1)$$

where $\eta_t > 0$ is the learning rate, and $\nabla\mathcal{L}_t(w)$ is the stochastic gradient of the negative log-posterior

$$\mathcal{L}_t(w) = \frac{|S_t|}{|S|} \sum_i \frac{w_i^2}{2\sigma_i^2} - \sum_{(x,a,y)\in S_t} \log P(\phi(x,a,y)|x)$$

restricted to batch S_t. The second term of the stochastic gradient involves only biphrases whose source sides match the source sentences in the batch. Though the gradient of the regularizer is non-zero for all non-zero biphrase features, the updates of features that are not active in the second term of the gradient can be postponed until they become active again. Due to feature sparsity, the number of features that are active in a small batch is small, and thus also the updates are sparse. Hence, it is possible to handle even feature vectors that do not fit into memory.

Another advantage of the stochastic gradient method is that many processes can apply updates (1) to a weight vector asynchronously in parallel. We have implemented two strategies for dealing with this. The simpler one is to store the weight vector in a database that takes care of the necessary concurrency control. This way, no process needs to store the entire weight vector in memory. The downside is that all training processes must be able to `mmap()` to the common file-system due to limitations in the underlying Berkeley DB database system. We have also implemented a client-server architecture in which a server process stores w in memory and manages read and update requests to its components that come from training clients. In this approach, the degree of parallelism is limited only by the number of available machines and server capacity. The server could be further distributed for managing models that do not fit into the memory of a single server.

3.4.3 Computing gradients etc

The computationally most challenging part in learning the model parameters is computing $\nabla \log P(\phi(x,a,y)|x)$, i.e., the vector of differences between the observed occurrence counts of biphrase features in (x, a, y) and their conditional expectations under the current model parameters.

The conditional feature expectations can be computed by a dynamic programming procedure similar to the one used in training conditional random fields. We combine the results of a left-to-right and right-to-left dynamic programming

sweep over x. In the left-to-right sweep, we have for each biphrase feature $(x', a', y'), i$ a state $s_{(x',a',y'),i}$ for translations starting from the beginning of x and ending in an occurrence of the biphrase (x', a', y') at source position i. This state records the contribution of all partial translations whose right-most active biphrase feature on the source side is $(x', a', y'), i$ to the conditional expectation of feature $(x', a', y'), i$ (in log scale). The score for $s_{\text{empty},0} = 0$, and $s_{(x',a',y'),i}$ is obtained from the recurrence

$$\text{score}\left(s_{(x',a',y'),i}\right) = \sum_{(x'',a'',y''),i'' \in A((x',a',y'),i)} \Big[\text{score}\left(s_{(x'',a'',y''),i''}\right) + \sum_{(x''',a''',y'''),i''' \in B} w_{(x''',a''',y'''),i'''}\Big]$$

Here, $A((x', a', y'), i)$ is the set of predecessor states of $s_{(x',a',y'),i}$ and includes all states $s_{(x'',a'',y''),i''}$ such that a proper suffix of $(x'', a'', y''), i''$ (i.e., a biphrase whose source and target are proper suffices of x'' and y'' respectively) is equal to a prefix of $(x', a', y'), i$. As a special case, A includes all states $s_{(x'',a'',y''),i''}$ for which $(x'', a'', y''), i''$ ends before or at position i. This takes care of translation paths that leave some words in x untranslated. Since the starting position of a proper suffix of a biphrase is always after the biphrase's original starting position, going through the states in order of increasing i guarantees that the scores for biphrases in $A((x', a', y'), i)$ are available when computing score $\left(s_{(x',a',y'),i}\right)$.

The set B that depends on $((x', a', y'), i)$ and $((x'', a'', y''), i'')$ is defined by the formula

$$B = \text{sub}\left((x', a', y'), i\right) \setminus \text{sub}\left((x'', a'', y''), i''\right),$$

where sub $((x', a', y'), i)$ denotes the set of sub-biphrases of $(x', a', y'), i$ (including the biphrase $(x', a', y'), i$ itself). Thus, summing over the weights of biphrases in B adds the contribution of features introduced by extending translation paths ending in $(x'', a'', y''), i''$ by $(x', a', y'), i'$.

From the right-to-left dynamic programming, we get analogously the contribution of right-to-left partial translations whose left-most active biphrase is $(x', a', y'), i$. The partition function used for normalizing the expectations can be obtained as a side product of either of the sweeps.

In conditional random fields, the (unnormalized) expectations for the feature can be obtained by multiplying the scores of the states corresponding to the same feature in the left-to-right and right-to-left dynamic programming memories. In our case, combining the two values stored in the states for a feature $(x', a', y'), i$ only gives the contribution of the translation paths where $(x', a', y'), i$ is active but not covered by any longer biphrase that extends $(x', a', y'), i$ both left and right. To include the contribution of the remaining translation paths, we need to go through states corresponding to super-biphrases of $(x', a', y'), i$. Special care has to be taken in order to include the contribution of all feature vectors in which such super-biphrases are active exactly once. An efficient way to do this is to process the states for super-biphrases in topological order with respect to biphrase inclusion, and to include only the contributions of states for super-biphrases that extend the previously included states both left and right.

Another complication in the dynamic programming is that a biphrase can extend the source side of another overlapping biphrase to the right, but the target side to the left, or visa versa. Such overlaps are not directly covered by our dynamic programming. To deal with them, we construct new virtual biphrases that correspond to the results of such overlaps in a pre-processing step. The number of such virtual combinations can in theory grow exponentially, but in practice only a small number of virtual biphrases seems to suffice.

3.5 Prediction with translation model alone

Prediction is done in two phases. First, we find (by a dynamic programming procedure similar to the one outlined in Section 3.4.3) the highest probability feature vector $\hat{\phi}(x)$ defined by

$$\hat{\phi}(x) = \underset{\phi \in \Phi_x \,:\, x \text{ covered by biphrases in } \phi}{\arg\max} P(\phi|x).$$

Note that we restrict the search to feature vectors that cover the whole of x, i.e., to feature vectors $\phi(x, a, y)$ in which each word in x is covered by at least one active aligned biphrase (x', a', y'). This forces the system to translate all words in the source sentence even if the translation model predicts that none of the translations are very likely.

To translate words that are not covered by any aligned biphrase feature in the model, we use the following strategy: If the word is found from an

optional out-of-vocabulary dictionary, we use the translation from the dictionary, and otherwise resort to an implicit zero weight aligned biphrase that copies the input word to the output as is. In our experiments, the out-of-vocabulary dictionary is constructed from the word translations that occur once in the training data, so the out-of-vocabulary dictionary only compensates for the word translations lost in phrase table pruning. If available, a real dictionary could be used as well.

The second step in predicting a translation is solving the pre-image problem, i.e., constructing a translation y from the predicted feature vector $\hat{\phi}(x)$. Since $\hat{\phi}(x) \in \Phi_x$, there always exists an alignment a and a translation y such that all the aligned biphrases in $\hat{\phi}(x)$ occur in $\phi(x, a, y)$, but the a and y may not be unique. We choose the a and y given by concatenating the target sides of the biphrases active in $\hat{\phi}(x)$ in the order induced by their positions in the source sentence. Thus, there is no phrase-level reordering, and the fluency of the target language output is induced by the phrase overlaps only.

3.6 Predicting with an integrated LM

The prediction strategy outlined in the previous section is simple and conceptually clean. However, biphrase overlaps alone may not be enough to enforce fluent output, especially given that bilingual data is typically more scarce than monolingual data. Also, the lack of a reverse translation model means the system is unable to identify phrase extraction errors in which rarely seen source phrases are translated to common target phrases by chance.

To address these shortcomings, we augment the translation model with the following additional features that have been observed to enhance translation quality in other SMT systems.

1. **Language model:** $\log P(y)$, where $P(y)$ is given by a smoothed n-gram language model
2. **Lexical translation model (reverse direction):** $\log P(x|y, a)$ given by a word-level reverse translation model
3. **Translation length:** number of words in y
4. **Distortion:** number of source words in phrases with swapped translations

The final score driving the translation process is given by a linear combination of the translation model score $\log P(\phi(x, a, y)|x)$ and these features. Besides the translation model, the language model feature is clearly the most influential, while the lexical translation feature has only a minor positive effect on translation quality.

We use an approximate dynamic programming variant of the commonly used beam search procedure to find the highest scoring candidate translation. We compute the translation model log-probability $\log P(\phi(x, a, y)|x)$ incrementally while building up the corresponding candidate translation y and word alignment a from left to right. We allow phrase-level distortions given by swapping the order of translations of consecutive non-overlapping source phrases. Unlike in `Moses`, our beam search is structured around state transitions, not around states. This means that we apply each biphrase (state transition) simultaneously to all applicable partial translations (states). This strategy is in our experience more efficient, does not rely on future score estimates, and is implementationally very similar to the dynamic programming procedures that we use in training the model parameters and in prediction with a language model alone.

The weights of the features are tuned by optimizing the BLEU score of development set translations with amoeba search. This simplistic strategy is feasible given our system's fast translation speed, and extends easily to cover non-linear feature combinations. The reason for using amoeba is that it is simpler to implement — we do not believe amoeba yields any better values for the parameters in the end.

4 Experiments

4.1 Experimental setup

Our experiments are on the Europarl translation tasks following the setup used in the shared translation task of the ACL 2008 Third Workshop on Statistical Machine Translation (Callison-Burch et al., 2008), and on the French-to-English translation task of the EACL 2009 Fourth Workshop on Statistical Machine Translation (Callison-Burch et al., 2009). The size of the Europarl training corpora is about 1M sentence pairs per language pair, while the larger GigaFrEn corpus contains about 22M sentence pairs. The corpora were used for biphrase extraction and translation model training. Decoder feature weights were tuned on the provided development sets. In case of Europarl, language models were trained on the target sides of

	es-en	en-es	fr-en	en-fr	de-en	en-de	time
Sinuhe	31.38	30.94	31.50	28.91	25.03	19.26	338.0
Moses	32.18	31.88	32.63	29.92	27.30	20.57	3729.5
$\text{Sinuhe}_{\text{trans}}$	29.14	27.12	28.74	26.06	22.38	17.14	44.2
$\text{Moses}_{\text{trans}}$	24.32	22.75	23.84	21.22	19.62	13.59	1321.5

Table 1: Left: The translation quality of the SMT systems as measured by the BLEU score. Translations were detokenized but not recased before evaluating their quality against lowercased reference translations by the `mteval-v11b.pl` script. Right: Average total translation time in seconds.

the bilingual corpora. In the GigaFrEn experiments we used the provided monolingual news domain data. All data was tokenized and lowercased using the tools in the Moses distribution.

We experimented with four translation systems: $\text{Sinuhe}_{\text{trans}}$, Sinuhe, $\text{Moses}_{\text{trans}}$, and Moses. $\text{Sinuhe}_{\text{trans}}$ uses only the translation model in producing translations (see Section 3.5), while the full system Sinuhe uses also a language model and some additional features (see Section 3.6). As a baseline, we used the Moses translation system, which is known to be very competitive on the Europarl translation tasks as evidenced by the University of Edinburgh entries in the translation challenge (Callison-Burch et al., 2008). The other comparison point $\text{Moses}_{\text{trans}}$ was obtained from Moses by disabling distortions and setting setting the weights of all features except the forward translation model to 0. By comparing $\text{Sinuhe}_{\text{trans}}$ and $\text{Moses}_{\text{trans}}$, we hope to indirectly compare the performance of the underlying translation models. A more direct comparison was not possible as it is not feasible to normalize the "probabilities" predicted by the Moses translation model.

4.1.1 Training the models

We trained Moses exactly as suggested in (Callison-Burch et al., 2008), except that we used the `-unk` option for SRILM in training the language models (both for Sinuhe and Moses). The translation model for Sinuhe (and $\text{Sinuhe}_{\text{trans}}$) was built from the phrases extracted by Moses as described in Section 3.1. We chose $\alpha = 1.0$ and set batch size to 1. The learning rate was initially set to 0.1, and decayed proportional to $1/t$ after 2M or 100M iterations of training for Europarl and GigaFrEn tasks, respectively. These choices may not be optimal as we did not experiment with other choices yet. In case of Europarl, training was run for 70-100M iterations using the Berkeley DB based distribution strategy (4 CPU cores per language pair). This took 10 days. For GigaFrEn, we used the client-server architecture, and trained the model for 620M stochastic gradient iterations on about 200 CPUs. This took 2 days, which is a lot less than the time needed to run (parallel) Giza on this data. The number of biphrase features in Sinuhe's model was 2-4 million on the Europarl tasks, and about 95 million on the GigaFrEn task.

The decoder parameters for Sinuhe were tuned on the development sets by amoeba, and for Moses by MERT. As both amoeba and MERT try to solve the same optimization problem, we believe the difference in optimization methods has little influence on the results.

4.1.2 Translation results

Europarl tasks The systems were tested on the 2000 sentence Europarl domain development test sets provided for the shared translation task (Callison-Burch et al., 2008). The resulting BLEU scores and total translation times averaged over the datasets are reported in Table 1. While Moses has the highest BLEU score for all the language pairs, the BLEU score for Sinuhe is worse by only at most 1.31 BLEU points except on the de-en task, where the difference is 2.27. $\text{Sinuhe}_{\text{trans}}$ is clearly inferior to Sinuhe but equally clearly superior to $\text{Moses}_{\text{trans}}$.

It takes less than a minute to translate the development test set by the fastest system $\text{Sinuhe}_{\text{trans}}$. The slowest system Moses needs around an hour for the same task. Memory usage follows a similar pattern. For example, Sinuhe requires roughly one tenth of the memory used by Moses. Thus, in terms of resource usage, $\text{Sinuhe}_{\text{trans}}$ and Sinuhe seem clearly superior to Moses. The quantitative results would change if the systems' parameters were optimized for speed rather than quality, but the differences

are so clear that the general pattern would probably remain the same. For example, Moses with no distortion is still clearly slower than Sinuhe.

GigaFrEn task The fr-en model was tested on the 2525 sentence news domain test data used in the preliminary evaluation of the translation challenge results. The BLEU scores for Sinuhe and Moses were 26.32 and 26.98, respectively. The total translation time was 13m 50s with Sinuhe, whereas Moses needed 82m 28s. Thus, the pattern that was observed on Europarl tasks is repeated here: The translation quality of Moses is slightly better, but Sinuhe is significantly faster. Surprisingly, both Sinuhe and Moses fare well in comparison to the participants of the actual challenge: According to the preliminary results on the same test data we used (Koehn, 2009), the Moses baseline would have been beaten only by Google, and Sinuhe would have been sixth among the 23 participating systems with a difference of only 0.57 BLEU points to the second best entry. A partial explanation for the good relative performance could be that the challenge participants had only a week to train their models on the full version of GigaFrEn data, so they may not have had time to take full advantage of it. On the other hand, many of the top ranked systems relied on external resources that were not available for us.

Based on an informal human evaluation of the outputs of Sinuhe and Moses, it looks like the translations of Sinuhe are slightly more accurate in conveying the meaning of the original sentences, but especially the translations of long rare expressions (e.g., multi-word names of institutions) are less fluent. This hints that the parameters for (rare) biphrases may have been regularized too heavily — it looks like Sinuhe is underfitting rather than overfitting. We will conduct more experiments to see how much the translation quality can be improved by a better choice of α or by using a different prior for regularization. Of course, there is room for tuning elsewhere, too. For example, it would be a surprise if the phrase extraction pipeline that has been optimized for Moses would be optimal for Sinuhe.

5 Conclusions

In this paper, we have shown that phrase-based SMT can be viewed as an instance of structural prediction. The word alignment and phrase extraction heuristics serve as a strategy for feature extraction, and the translation task can be modelled as a structural prediction problems over these features. Our methods scale to large corpora and are fast at predicting translations. While speed is not the primary goal, the faster translation times may be a key to success in applications where the amount of text that needs to be translated is large. In terms of BLEU scores, the results do not improve the state-of-the-art so far. However, fine-tuning the standard phrase-based approach over the years has increased its performance significantly, and we see no reason why the same would not happen with the proposed approach, especially if the model is augmented with additional features like gapped biphrases and biphrases over POS tags.

The fact that our translation model is a properly normalized conditional probability distribution opens up many new possibilities. For instance, instead of predicting translations, it is possible to efficiently compute the expected number of times each word would appear in them. Such output might be useful, e.g., if the translations are to be post-processed by models relying on bag-of-words representation. Another research direction we are currently looking into is training the proposed translation model in the reverse direction, and then predicting translations using the noisy channel approach, i.e., by maximizing $P(x|y)P(y)$. The key difference to previous work here is that since $P(x|y)$ is properly normalized, the noisy channel approach would not in our case suffer from the potentially negative effects caused by ignoring the normalizer that depends on y. Besides being a viable (though computationally demanding) alternative criterion for predicting translations, the noisy channel approach could easily be used for, e.g., reranking n-best lists and for system combination.

Acknowledgments

This work has been partially funded by the SMART EU project. We wish to thank the anonymous reviewers for their constructive comments and especially for pointing out the related paper (Blunsom and Osborne, 2008) that we had missed. Special thanks to Vladimir Poroshin for all the bugs he found in beta-testing and to Esther Galbrun for her comments on a draft version of this paper.

References

Phil Blunsom and Miles Osborne. 2008. Probabilistic inference for machine translation. In *EMNLP*.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *ACL*.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Josh Schroeder, and Cameron Shaw Fordyce. 2008. ACL 2008 third workshop on statistical machine translation. `http://www.statmt.org/wmt08`.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. EACL 2009 fourth workshop on statistical machine translation. `http://www.statmt.org/wmt09`.

Stanley Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for ME models. *IEEE Transactions Speech and Audio Processing*, 8(1):37–50.

John DeNero and Dan Klein. 2008. The complexity of phrase alignment problems. In *ACL*.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *Workshop on SMT at NAACL*.

John DeNero, Alex Bouchard, and Dan Klein. 2008. Sampling alignment structure under a bayesian translation model. In *EMNLP*.

Matti Kääriäinen. 2009. Sinuhe source code distribution (v1.2). Website. `http://www.cs.helsinki.fi/u/mtkaaria/sinuhe`.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, and Chris Callison-Burch et al. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

Philipp Koehn. 2009. BLEU/NIST scores for submissions. `http://groups.google.com/group/WMT09/browse_thread/thread/bfbce7b219648a4c`.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Percy Liang, Alexandre Bouchard-Cote, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *ACL*.

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

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

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *EMNLP*, pages 20–28.

Cristoph Tillmann and Tong Zhang. 2007. A block bigram prediction model for statistical machine translation. *ACM Transacions on Speech and Language Processing*, 4(3).

S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. 2006. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML*.

Fast Translation Rule Matching for Syntax-based Statistical Machine Translation

Hui Zhang[1, 2] **Min Zhang**[1] **Haizhou Li**[1] **Chew Lim Tan**[2]
[1]Institute for Infocomm Research [2]National University of Singapore
zhangh1982@gmail.com {mzhang, hli}@i2r.a-star.edu.sg tancl@comp.nus.edu.sg

Abstract

In a linguistically-motivated syntax-based translation system, the entire translation process is normally carried out in two steps, translation rule matching and target sentence decoding using the matched rules. Both steps are very time-consuming due to the tremendous number of translation rules, the exhaustive search in translation rule matching and the complex nature of the translation task itself. In this paper, we propose a hyper-tree-based fast algorithm for translation rule matching. Experimental results on the NIST MT-2003 Chinese-English translation task show that our algorithm is at least 19 times faster in rule matching and is able to help to save 57% of overall translation time over previous methods when using large fragment translation rules.

1 Introduction

Recently linguistically-motivated syntax-based translation method has achieved great success in statistical machine translation (SMT) (Galley et al., 2004; Liu et al., 2006, 2007; Zhang et al., 2007, 2008a; Mi et al., 2008; Mi and Huang 2008; Zhang et al., 2009). It translates a source sentence to its target one in two steps by using structured translation rules. In the first step, which is called translation rule matching step, all the applicable[1] translation rules are extracted from the entire rule set by matching the source parse tree/forest. The second step is to decode the source sentence into its target one using the extracted translation rules. Both of the two steps are very time-consuming due to the exponential number of translation rules and the complex nature of machine translation as an NP-hard search problem (Knight, 1999). In the SMT research community, the second step has been well studied and many methods have been proposed to speed up the decoding process, such as node-based or span-based beam search with different pruning strategies (Liu et al., 2006; Zhang et al., 2008a, 2008b) and cube pruning (Huang and Chiang, 2007; Mi et al., 2008). However, the first step attracts less attention. The previous solution to this problem is to do exhaustive searching with heuristics on each tree/forest node or on each source span. This solution becomes computationally infeasible when it is applied to packed forests with loose pruning threshold or rule sets with large tree fragments of large rule height and width. This not only overloads the translation process but also compromises the translation performance since as shown in our experiments the large tree fragment rules are also very useful.

To solve the above issue, in this paper, we propose a hyper-tree-based fast algorithm for translation rule matching. Our solution includes two steps. In the first step, all the translation rules are re-organized using our proposed hyper-tree structure, which is a compact representation of the entire translation rule set, in order to make the common parts of translation rules shared as much as possible. This enables the common parts of different translation rules to be visited only once in rule matching. Please note that the first step can be easily done off-line very fast. As a result, it does not consume real translation time. In the second step, we design a recursive algorithm to traverse the hyper-tree structure and the input source forest in a top-down manner to do the rule matching between them. As we will show later, the hyper-tree structure and the recursive algorithm significantly improve the speed of the rule matching and the entire translation process compared with previous methods.

With the proposed algorithm, we are able to carry out experiments with very loose pruning

[1] Given a source structure (either a parse tree or a parse forest), a translation rule is applicable if and only if the left hand side of the translation rule exactly matches a tree fragment of the given source structure.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1037–1045,
Singapore, 6-7 August 2009.

thresholds and larger tree fragment rules efficiently. Experimental results on the NIST MT-2003 Chinese-English translation task shows that our algorithm is 19 times faster in rule matching and is able to save 57% of overall translation time over previous methods when using large fragment translation rules with height up to 5. It also shows that the larger rules with height of up to 5 significantly outperforms the rules with height of up to 3 by around 1 BLEU score.

The rest of this paper is organized as follows. Section 2 introduces the syntax-based translation system that we are working on. Section 3 reviews the previous work. Section 4 explains our solution while section 5 reports the experimental results. Section 6 concludes the paper.

2 Syntax-based Translation

This section briefly introduces the forest/tree-based tree-to-string translation model which serves as the translation platform in this paper.

2.1 Tree-to-string model

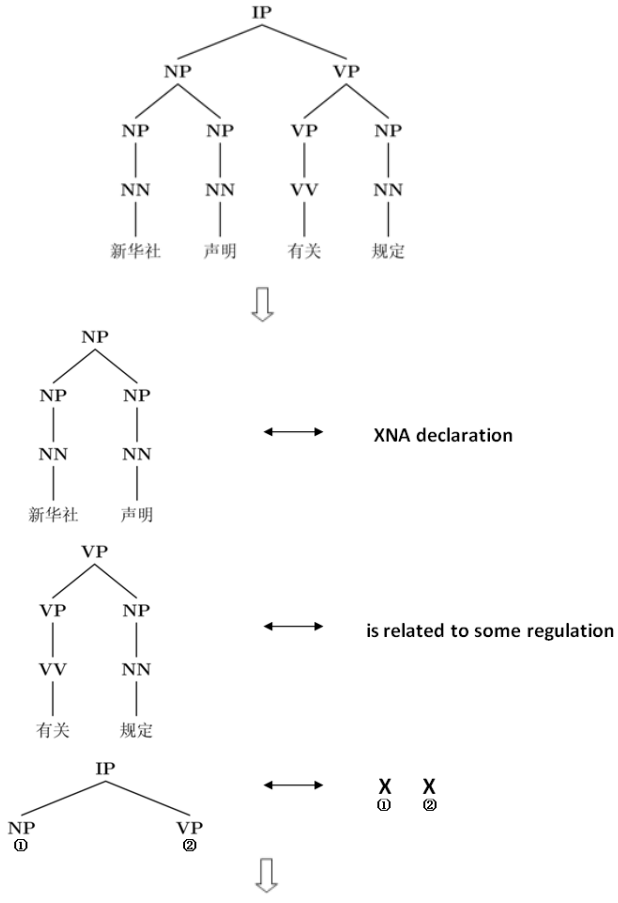


Figure 1. A tree-to-string translation process.

The tree-to-string model (Galley et al. 2004; Liu et al. 2006) views the translation as a structure mapping process, which first breaks the source syntax tree into many tree fragments and then maps each tree fragment into its corresponding target translation using translation rules, finally combines these target translations into a complete sentence. Fig. 1 illustrates this process. In real translation, the number of possible tree fragment segmentations for a given input tree is exponential in the number of tree nodes.

2.2 Forest-based translation

To overcome parse error for SMT, Mi and Huang (2008) propose forest-based translation by using a packed forest instead of a single syntax tree as the translation input. A packed forest (Tomita 1987; Klein and Manning, 2001; Huang and Chiang, 2005) is a compact representation of many possible parse trees of a sentence, which can be formally described as a triple $< V, E, S >$, where V is the set of non-terminal nodes, E is the set of hyper-edges and S is a sentence represented as an ordered word sequence. A hyper-edge in a packed forest is a group of edges in a tree which connects a father node to all its children nodes, representing a CFG-based parse rule. Fig. 2 is a packed forest incorporating two parse trees T1 and T2 of a sentence as shown in Fig. 3 and Fig. 4. Given a hyper-edge e, let h be its father node, then we say that e is attached to h.

A non-terminal node in a packed forest can be represented as "label [start, stop]", where "label" is its syntax category and "[start, stop]" is the range of words it covers. For example, the node in Fig. 5 pointed by the dark arrow is labelled as "NP[3,4]", where NP is its label and [3,4] means that it covers the span from the 3rd word to the 4th word. In forest-based translation, rule matching is much more complicated than the tree-based one.

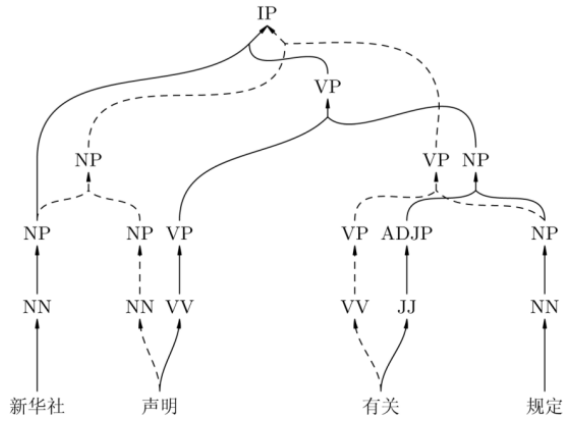


Figure 2. A packed forest

Zhang et al. (2009) reduce the tree sequence problem into tree problem by introducing virtual node and related forest conversion algorithms, so

the algorithm proposed in this paper is also applicable to the tree sequence-based models.

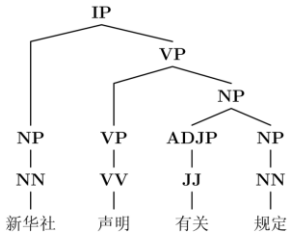


Figure 3. Tree 1 (T1)

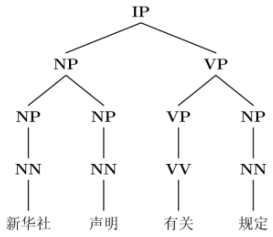


Figure 4. Tree 2 (T2)

3 Matching Methods in Previous Work

In this section, we discuss the two typical rule matching algorithms used in previous work.

3.1 Exhaustive search by tree fragments

This method generates all possible tree fragments rooted by each node in the source parse tree or forest, and then matches all the generated tree fragments against the source parts (left hand side) of translation rules to extract the useful rules (Zhang et al., 2008a).

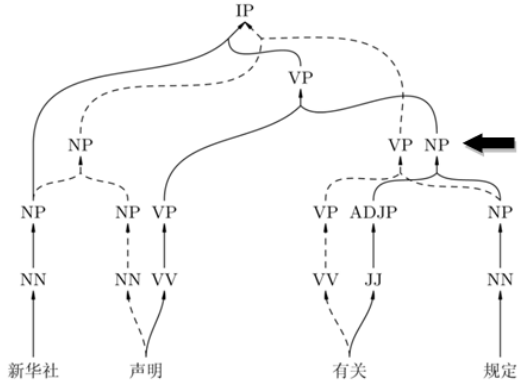


Figure 5. Node NP[3,4] in packed forest

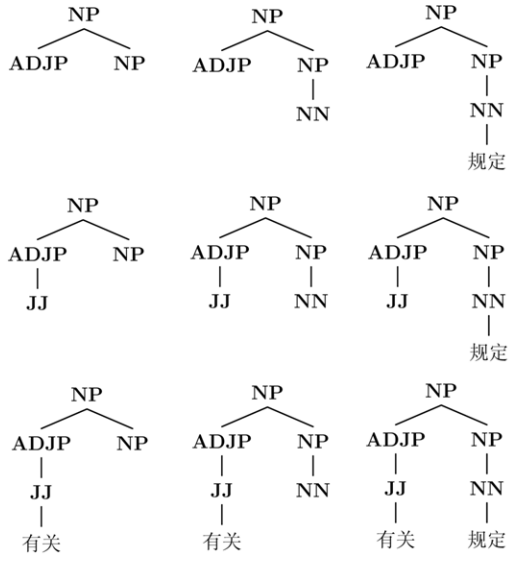


Figure 6. Candidate fragments on NP[3,4]

For example, if we want to extract useful rules for node NP[3,4] in Fig 5, we have to generate all the tree fragments rooted at node NP[3,4] as shown in Fig 6, and then query each fragment in the rule set. Let h be a node in the packed forest, $f(h)$ represents the number of possible tree fragments rooted at node h, then we have:

$$f(h) = \begin{cases} 0, & if\ h\ is\ a\ leaf\ node \\ \sum_{\substack{e\ is\ a\ hyper-edge \\ attached\ to\ h}} \prod_{\substack{c_i\ is\ the \\ i^{th}\ children \\ node\ in\ e}} (1 + f(c_i)), & otherwise \end{cases}$$

The above equation shows that the number of tree fragments is exponential to the span size, the height and the number of hyper-edges it covers. In a real system, one can use heuristics, e.g. the maximum number of nodes and the maximum height of fragment, to limit the number of possible fragments. However, these heuristics are very subjective and hard to optimize. In addition, they may filter out some "good" fragments.

3.2 Exhaustive search by rules

This method does not generate any source tree fragments. Instead, it does top-down recursive matching from each node one-by-one with each translation rule in the rule set (Mi and Huang 2008).

For example, given a translation rule with its left hand side as shown in Fig. 7, the rule matching between the given rule and the node IP[1,4] in Fig. 2 can be done as follows.

1. Decompose the left hand side of the translation rule as shown in Fig. 7 into a sequence of hyper-edges in top-down, left-to-right order as follows:

IP => NP VP; NP => NP NP; NP => NN;
NN => 声明

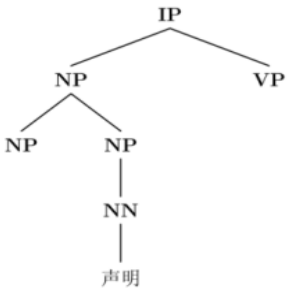


Figure 7. The left hand side of a rule

2. Pattern match these hyper-edges(rule) one-by-one in top-down left-to-right order from node IP[1,4]. If there is a continuous path in the forest matching all of these hyper-edges in order, then we can say that the rule is useful and matchable

with the tree fragment covered by the continuous path. The following illustrates the matching steps:

1. Match hyper-edge "IP => NP VP" with node IP[1,4]. There are two hyper-edges in the forest matching it: "IP[1,4] => NP[1,1] VP[2,4]" and "IP[1,4] => NP[1,2] VP [3,4]", which generates two candidate paths.

2. Since hyper-edge "NP => NP NP" fails to match NP[1,1], the path initiated with "IP[1,4] => NP[1,1] VP[2,4]" is pruned out.

3. Since there is a hyper-edge "NP[1,2] => NP[1,1] NP[2,2]" matching "NP => NP NP" on NP[1,2], then continue for further matching.

4. Since "NP=>NN" on NP[2,2] matches "NP[2,2] => NN[2,2]", then continue for further matching.

5. "NN=>声明" on NN[2,2] matches "NN[2,2] =>声明" and it is the last hyper-edge in the input rules. Finally, there is one continuous path successfully matching the left hand side of the input rule.

This method is able to avoid the exponential problem of the first method as described in the previous subsection. However, it has to do one-by-one pattern matching for each rule on each node. When the rule set is very large (indeed it is very large in the forest-based model even with a small training set), it becomes very slow, and even much slower than the first method.

4 The Proposed Hyper-tree-based Rule Matching Algorithm

In this section, we first explain the motivation why we re-organize the translation rule sets, and then elaborate how to re-organize the translation rules using our proposed hyper-tree structure. Finally we discuss the top-down rule matching algorithm between forest and hyper-tree.

4.1 Motivation

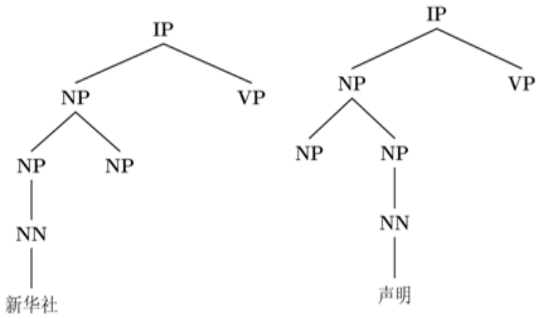


Figure 8. Two rules' left hand side

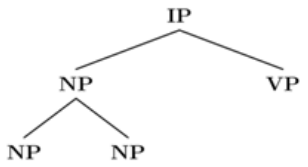


Figure 9. Common part of the two rules' left hand sides in Figure 8

Fig. 9 shows the common part of the left hand sides of two translation rules as shown in Fig. 8. In previous rule matching algorithm, the common parts are matched as many times as they appear in the rule set, which reduces the rule matching speed significantly. This motivates us to propose the hyper-tree structure and the rule matching algorithm to make the common parts shared by multiple translation rules to be visited only once in the entire rule matching process.

4.2 Hyper-node, hyper-path and hyper-tree

A hyper-tree is a compact representation of a group of tree translation rules with common parts shared. It consists of a set of hyper-nodes with edges connecting different hyper-nodes into a big tree. A hyper-tree is constructed from the translation rule sets in two steps:

1) Convert each tree translation rule into a hyper-path;
2) Construct the hyper-tree by incrementally adding each individual hyper-path into the hyper-tree.

A tree rule can be converted into a hyper-path without losing information. Fig. 10 demonstrates the conversion process:

1) We first fill the rule tree with virtual nodes to make all its leaves have the same depth to the root;
2) We then group all the nodes in the same tree level to form a single hyper-node, where we use a comma as a delimiter to separate the tree nodes with different father nodes;
3) A hyper-path is a set of hyper-nodes linked in a top-down manner.

The commas and virtual nodes ε are introduced to help to recover the original tree from the hyper-path. Given a tree node in a hyper-node, if there are n commas before it, then its father node is the $(n+1)^{th}$ tree node in the father hyper-node. If we could find father node for each node in hyper-nodes, then it is straightforward to recover the original tree from the hyper-path by just adding the edges between original father and children nodes except the virtual node ε.

After converting each tree rule into a hyper-path, we can organize the entire rule set into a big hyper-tree as shown in Figure 11. The concept of hyper-path and hyper-tree could be viewed as an extension of the "prefix merging" ideas for CFG rules (Klein and Manning 2001).

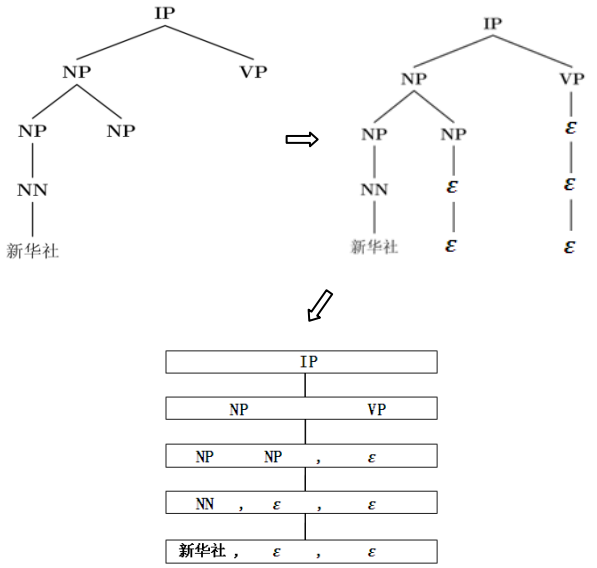


Figure 10. Convert tree to hyper-path

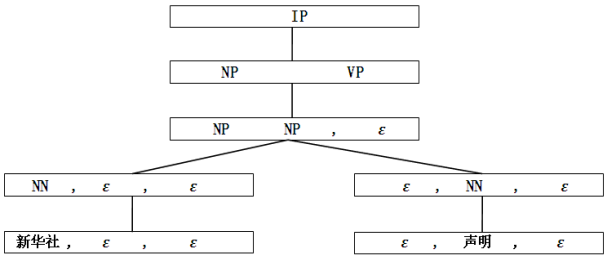


Figure 11. A hyper-tree example

Algorithm 1 shows how to organize the rule set into a big hyper-tree. The general process is that for each rule we convert it into a hyper-path and then add the hyper-path into a hyper-tree incrementally. However, there are many different hyper-trees generated given a big rule set. We then introduce a TOP label as the root node to link all the individual hyper-trees to a single big hyper-tree. Algorithm 2 shows the process of adding a hyper-path into a hyper-tree. Given a hyper-path, we do a top-down matching between the hyper-tree and the input hyper-path from root hyper-node until a leaf hyper-node is reached or there is no matching hyper-node at some level found. Then we add the remaining unmatchable part of the input hyper-path as the descendants of the last matchable hyper-node.

Please note that in Fig. 10 and Fig. 11, we ignore the target side (right hand side) of translation rules for easy discussion. Indeed, we can easily represent all the complete translation rules (not only left hand side) in Fig. 11 by simply adding the corresponding rule target sides into each hyper-node as done by line 5 of Algorithm 1.

Any hyper-path from the root to any hyper-node (not necessarily be a leaf of the hyper-tree) in a hyper-tree can represent a tree fragment. As a result, the hyper-tree in Fig. 11 can represent up to 6 candidate tree fragments. It is easy to understand that the maximum number of tree fragments that a hyper-tree can represent is equal to the number of hyper-nodes in it except the root. It is worth noting that a hyper-node in a hyper-tree without any target side rule attached means there is no translation rule corresponding to the tree fragment represented by the hyper-path from the root to the current hyper-node. The compact representation of the rule set by hyper-tree enables a fast algorithm to do translation rule matching.

Algorithm 1. *Compile rule set into hyper-tree*
Input: *rule set*
Output: *hyper-tree*

1. Initialize hyper-tree as a TOP node
2. **for** each rule in rule set **do**
3. Convert the left hand side tree to a hyper-path p
4. Add hyper-path p into hyper-tree
5. Add rule's right hand side to the leaf hyper-node of a hyper-path in the hyper-tree
6. **end for**

Algorithm 2. *Add hyper-path into hyper-tree*
Input: *hyper-path p and hyper-tree t*
Notation:
h: the height of hyper-path p
p(i) : the hyper-node of ith level (top-down) of p
TN: the hyper-node in hyper-tree
Output: *updated hyper-tree t*

1. Initialize TN as TOP
2. **for** i := 1 to h **do**
3. **if** there is a child c of TN has the same label as p(i) **then**
4. TN := c
5. **else**
6. Add a child c to TN, label c as p(i)
7. TN := c

4.3 Translation rule matching between forest and hyper-tree

Given the source parse forest and the translation rules represented in the hyper-tree structure, here we present a fast matching algorithm to extract so-called useful translation rules from the entire rule set in a top-down manner for each node of the forest.

As shown in Algorithm 3, the general process of the matching algorithm is as follows:

Algorithm 3. *Rule matching on one node*
Input: *hyper-tree T, forest F, and node n*
Notation:
FP: a pair <FNS, TN>, FNS is the frontier nodes of matched tree fragment,
TN is the hyper-tree node matching it
SFP: the queue of FP
Output: *Available rules on node n*

```
1.  if there is no child c of TOP having the same label as n
    then
2.      Return failure.
3.  else
4.      Initialize FP as <{n},c> and put it into SFP
5.      for each FP in SFP do
6.          SFP ← PropagateNextLevel(FP.FNS, FP.TN)
7.      for each FP in SFP do
8.          if the rule set attached to FP.TN is not empty
            then
9.              Add FP to result
```

Algorithm 4. *PropagateNextLevel*
Input: *Frontier node sequence FNS, hyper-tree node TN*
Notation:
CT: a child node of TN
the number of node sequence (separated by comma, see Fig 11) in CT is equal to the number of node in TN.
CT(i) : the ith node sequence in hyper-node CT
FNS(i): the ith node in FNS
TFNS: the temporary set of frontier node sequence
RFNS: the result set of frontier node sequence
FP: a pair of frontier node sequence and hyper-tree node
RFP: the result set of FP
Output: *RFP*

```
1.  for each child hyper-node CT of TN do
2.      for i:= 1 to the number of node sequence in CT do
3.          empty TFNS
4.          if CT(i) == ε then
5.              Add FNS(i) to TFNS.
6.          else
7.              for each hyper-edge e attached to FNS(i) do
8.                  if e.children match CT(i) then
9.                      Add e.children to TFNS
10.         if TFNS is empty then
11.             empty RFNS
12.             break
13.         else if i == 1 then
14.             RFNS := TFNS
15.         else
16.             RFNS := RFNS ⊗ TFNS
17.     for each FNS in RFNS do
18.         add <FNS, CT > into RFP
```

1) For each node *n* of the source forest if no child node of TOP in hyper-tree has the same label with it, it means that no rule matches any tree fragments rooted at the node *n* (i.e., no useful rules to be used for the node *n*) (line 1-2)

2) Otherwise, we match the sub-forest starting from the node *n* against a sub-hyper-tree starting from the matchable child node of TOP layer by layer in a top-down manner. There may be many possible tree fragments rooted at node *n* and each of them may have multiple useful translation rules. In our implementation, we maintain a data structure of FP = <FNS, TN> to record the currently matched tree fragment of forest and its corresponding hyper-tree node in the rule set, where FNS is the frontier node set of the current tree fragment and TN is the hyper-tree node. The data structure FP is used to help extract useful translation rules and is also used for further matching of larger tree fragments. Finally, all the FPs for the node *n* are kept in a queue. During the search, the queue size is dynamically increased. The matching algorithm terminates when all the FPs have been visited (line 5-6 and Algorithm 4).

3) In the final queue, each element (FP) of the queue contains the frontier node sequence of the matched tree fragment and its corresponding hyper-tree node. If the target side of a rule in the hyper-tree node is not empty, we just output the frontier nodes of the matched tree fragment, its root node *n* and all the useful translation rules for later translation process.

Algorithm 4 describes the detailed process of how to propagate the matching process down to the next level. <FNS, TN> is the current level frontier node sequence and hyper-tree node. Given a child hyper-node CT of TN (line 1), we try to find the group of next level frontier node sequence to match it (line 2-18). As shown in Fig 11, a hyper-node consists of a sequence of node sequence with comma as delimiter. For the i^{th} node sequence CT(i) in CT, If CT(i) is ε, that means FNS(i) is a leaf/frontier node in the matched tree fragment and thus no need to propagate to the next level (line 4-5). Otherwise, we try each hyper-edge e of FNS(i) to see whether its children match CT(i), and put the children of the matched hyper-edge into a temp set TFNS (line 7-9). If the temp set is empty, that means the current matching fails and no further expansion needs (line 10-12). Otherwise, we integrate current matched children into the final group of frontier node sequence (line 13-16) by Descartes Product (⊗). Finally, we construct all the <FNS, TN> pair for next level matching (line 17-18).

It would be interesting to study the time complexity of our Algorithm 3 and 4. Suppose the maximum number of children of each hyper-node in hyper-tree is N (line 1), the maximum number of node sequence in CT is M (line 2), the maximum number of hyper-edge in each node in packed forest is K (line 7), the maximum number of hyper-edge with same children representation in each node in packed forest is C (i.e. the maximum size of TFNS in line 16, and the maximum complexity of the Descartes Product in line 16

would be C^M), then the time complexity upper-bound of Algorithm 4 is $O(NM(K+C^M))$. For Algorithm 3, its time complexity is $O(RNM(K+C^M))$, where R is the maximum number of tree fragment matched in each node.

5 Experiment

5.1 Experimental settings

We carry out experiment on Chinese-English NIST evaluation tasks. We use FBIS corpus (250K sentence pairs) as training data with the source side parsed by a modified Charniak parser (Charniak 2000) which can output a packed forest. The Charniak Parser is trained on CTB5, tuned on 301-325 portion, with F1 score of 80.85% on 271-300 portion. We use GIZA++ (Och and Ney, 2003) to do *m-to-n* word-alignment and adopt heuristic "grow-diag-final-and" to do refinement. A 4-gram language model is trained on Gigaword 3 Xinhua portion by SRILM toolkit (Stolcke, 2002) with Kneser-Ney smoothing. We use NIST 2002 as development set and NIST 2003 as test set. The feature weights are tuned by the modified Koehn's MER (Och, 2003, Koehn, 2007) trainer. We use case-sensitive BLEU-4 (Papineni et al., 2002) to measure the quality of translation result. Zhang et al. 2004's implementation is used to do significant test.

Following (Mi and Huang 2008), we use viterbi algorithm to prune the forest. Instead of using a static pruning threshold (Mi and Huang 2008), we set the threshold as the distance of the probabilities of the n^{th} best tree and the 1^{st} best tree. It means the pruned forest is able to at least keep all the top n best trees. However, because of the sharing nature of the packed forest, it may still contain a large number of additional trees. Our statistic shows that when we set the threshold as the 100^{th} best tree, the average number of all possible trees in the forest is $1.2*10^5$ after pruning.

In our experiments, we compare our algorithm with the two traditional algorithms as discussed in section 3. For the "Exhaustive search by tree" algorithm, we use a bottom-up dynamic programming algorithm to generate all the candidate tree fragments rooted at each node. For the "Exhaustive search by rule" algorithm, we group all rules with the same left hand side in order to remove the duplicated matching for the same left hand side rules. All these settings aim for fair comparison.

5.2 Accuracy, speed vs. rule heights

We first compare the three algorithms' performance by setting the maximum rule height from 1 to 5. We set the forest pruning threshold to the 100^{th} best parse tree.

Table 1 compares the speed of the three algorithms. It clearly shows that the speed of both of the two traditional algorithms increases dramatically while the speed of our hyper-tree based algorithm is almost linear to the tree height. In the case of rule height of 5, the hyper-tree algorithm is at least 19 times (9.329/0.486) faster than the two traditional algorithms and saves 8.843(9.329 - 0.486) seconds in rule matching for each sentence on average, which contributes 57% (8.843/(9.329 + 6.21)) speed improvement to the overall translation.

H	Rule Matching			D
	Exhaustive by tree	Exhaustive by rule	Hyper-tree-based	
1	0.043	0.077	0.083	2.96
2	0.047	0.920	0.173	3.56
3	0.237	9.572	0.358	4.02
4	2.300	48.90	0.450	5.27
5	9.329	90.80	0.486	6.21

Table 1. Speed in seconds per sentence vs. rule height; "H" is rule height, "D" represents the decoding time after rule matching

Height	BLEU
1	0.1646
2	0.2498
3	0.2824
4	0.2874
5	**0.2925**
Moses	0.2625

Table 2. BLEU vs. rule height

Table 2 reports the BLEU score with different rule heights, where Moses, a state-of-the-art phrase-based SMT system, serves as the baseline system. It shows the BLEU score consistently improves as the rule height increases. In addition, one can see that the rules with maximum height of 5 are able to outperform the rules with maximum height of 3 by 1 BLEU score ($p<0.05$) and significantly outperforms Moses by 3 BLEU score ($p<0.01$). To our knowledge, this is the first time to report the performance of rules up to height of 5 for forest-based translation model.

We also study the distribution of the rules used in the 1-best translation output. The results are shown in Table 3; we could see something interesting that is as the rule height increases, the total number of rules with that height decreases, while the percentage of partial-lexicalized increases dramatically. And one thing needs to note is the percentage of partial-lexicalized rules with height of 1 is 0, since there is no partial-lexicalized rule with height of 1 in the rule set (the father node of a word is a pos tag node).

H	Total	Rule Type Percentage (%)		
		F	P	U
1	9814	76.58	0	23.42
2	5289	44.99	46.40	8.60
3	3925	18.39	77.25	4.35
4	1810	7.90	87.68	4.41
5	511	6.46	90.50	3.04

Table 3. statistics of rules used in the 1-best translation output, "F" means full-lexicalized, "P" means partial-lexicalized, "U" means unlexiclaizd.

5.3 Speed vs. forest pruning threshold

This section studies the impact of the forest pruning threshold on the rule matching speed when setting the maximum rule height to 5.

Threshold	Rule Matching		
	Exhaustive by tree	Exhaustive by rule	Hyper-tree-based
1	1.2	23.66	0.171
10	3.1	36.42	0.234
50	5.7	66.20	0.405
100	9.3	90.80	0.486
200	27.3	104.86	0.598
500	133.6	148.54	0.873

Table 4. Speed in seconds per sentence vs. forest pruning threshold

In Table 4, we can see that our hyper-tree based algorithm is the fastest among the three algorithms in all pruning threshold settings and even 150 times faster than both of the two traditional algorithms with threshold of 500th best. Table 5 shows the average number of parse trees embedded in a packed forest with different pruning thresholds per sentence. We can see that the number of trees increases exponentially when the pruning threshold increases linearly. When the threshold is 500th best, the average number of trees per sentence is $1.49*10^9$. However, even in this extreme case, the hyper-tree based algorithm is still capable of completing rule matching within 1 second.

Threshold	Number of Trees
1	1
10	32
50	5922
100	128860
200	$2.75*10^6$
500	$1.49*10^9$

Table 5. Average number of trees in packed forest with different pruning threshold.

5.4 Hyper-tree compression rate

As we describe in section 4.2, theoretically the number of tree fragments that a hyper-tree can represent is equal to the number of hyper-nodes in it. However, in real rule set, there is no guarantee that each tree fragment in the hyper-tree has corresponding translation rules. To gain insights into how effective the compact representation of the hyper-tree and how many hyper-nodes without translation rules, we define the compression rate as follows.

$$compression\ rate\ = \frac{the\ number\ tree\ fragments\ having\ trans\ rules}{the\ total\ number\ of\ tree\ fragments}$$

Table 6 reports the different statistics on the rule sets with different maximum rule heights ranging from 1 to 5. The reported statistics are the number of rules, the number of unique left hand side (since there may be more than one rules having the same left hand side), the number of hyper-nodes and the compression rate.

H	n_rules	n_LHS	n_nodes	c_rate
1	21588	10779	10779	100%
2	141632	51807	51903	99.8%
3	$1.73*10^6$	491268	494919	99.2%
4	$8.65*10^6$	2052731	2083296	98.5%
5	$1.89*10^7$	3966742	4043824	98.1%

Table 6. Statistics of rule set and hyper-tree. "H" is rule height, "n_rules" is the number of rules, "n_LHS" is the number of unique left hand side, "n_nodes" is the number of hyper-nodes in hyper-tree and "c_rate" is the compression rate.

Table 6 shows that in all the five cases, the compression rates of hyper-tree are all more than

98%. It means that almost all the tree fragments embedded in the hyper-tree have corresponding translation rules. As a result, we are able to use almost only one hyper-edge (i.e. only the frontier nodes of a tree fragment without any internal nodes) to represent all the rules with the same left hand side. This suggests that our hyper-tree is particularly effective in representing the tree translation rules compactly. It also shows that there are a lot of common parts among different translation rules.

All the experiments reported in this section convincingly demonstrate the effectiveness of our proposed hyper-tree representation of translation rules and the hyper-tree-based rule matching algorithm.

6 Conclusion

In this paper, we propose the concept of hyper-tree for compact rule representation and a hyper-tree-based fast algorithm for translation rule matching in a forest-based translation system. We compare our algorithm with two previous widely-used rule matching algorithms. Experimental results on the NIST Chinese-English MT 2003 evaluation data set show the rules with maximum rule height of 5 outperform those with height 3 by 1.0 BLEU and outperform MOSES by 3.0 BLEU. In the same test cases, our algorithm is at least 19 times faster than the two traditional algorithms, and contributes 57% speed improvement to the overall translation. We also show that in a more challenging setting (forest containing $1.49*10^9$ trees on average) our algorithm is 150 times faster than the two traditional algorithms. Finally, we show that the hyper-tree structure has more than 98% compression rate. It means the compact representation by the hyper-tree is very effective for translation rules.

References

Eugene Charniak. 2000. *A maximum-entropy inspired parser*. NAACL-00.

Michel Galley, Mark Hopkins, Kevin Knight and Daniel Marcu. 2004. *What's in a translation rule?* HLT-NAACL-04.

Liang Huang and David Chiang. 2005. *Better k-best Parsing*. IWPT-05.

Liang Huang and David Chiang. 2007. *Forest rescoring: Faster decoding with integrated language models*. ACL-07. 144–151

Dan Klein and Christopher D. Manning. 2001. *Parsing and Hypergraphs*. IWPT-2001.

Dan Klein and Christopher D. Manning. 2001. *Parsing with Treebank Grammars: Empirical Bounds, Theoretical Models, and the Structure of the Penn Treebank*. ACL - 2001. 338-345.

Kevin Knight. 1999. *Decoding Complexity in Word-Replacement Translation Models*. CL: J99-4005.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin and Evan Herbst. 2007. *Moses: Open Source Toolkit for Statistical Machine Translation.* ACL-07. 177-180. (poster)

Yang Liu, Qun Liu and Shouxun Lin. 2006. *Tree-to-String Alignment Template for Statistical Machine Translation.* COLING-ACL-06. 609-616.

Yang Liu, Yun Huang, Qun Liu and Shouxun Lin. 2007. *Forest-to-String Statistical Translation Rules.* ACL-07. 704-711.

Haitao Mi, Liang Huang, and Qun Liu. 2008. *Forest-based translation*. ACL-HLT-08. 192-199.

Haitao Mi and Liang Huang. 2008. *Forest-based Translation Rule Extraction*. EMNLP-08. 206-214

Franz J. Och. 2003. *Minimum error rate training in statistical machine translation.* ACL-03. 160-167.

Franz Josef Och and Hermann Ney. 2003. *A Systematic Comparison of Various Statistical Alignment Models*. Computational Linguistics. 29(1) 19-51

Kishore Papineni, Salim Roukos, ToddWard and Wei-Jing Zhu. 2002. *BLEU: a method for automatic evaluation of machine translation.* ACL-02.311-318.

Andreas Stolcke. 2002. *SRILM - an extensible language modeling toolkit.* ICSLP-02. 901-904.

Masaru Tomita. 1987. *An Efficient Augmented-Context-Free Parsing Algorithm*. Computational Linguistics 13(1-2): 31-46.

Hui Zhang, Min Zhang, Haizhou Li, Aiti Aw and Chew Lim Tan. 2009. *Forest-based Tree Sequence to String Translation Model.* ACL-IJCNLP-09.

Min Zhang, Hongfei Jiang, Ai Ti Aw, Jun Sun, Sheng Li and Chew Lim Tan. 2007. *A Tree-to-Tree Alignment-based Model for Statistical Machine Translation*. MT-Summit-07. 535-542.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, Sheng Li. 2008a. *A Tree Sequence Alignment-based Tree-to-Tree Translation Model.* ACL-HLT-08. 559-567.

Min Zhang, Hongfei Jiang, Haizhou Li, Aiti Aw, Sheng Li. 2008b. *Grammar Comparison Study for Translational Equivalence Modeling and Statistical Machine Translation.* COLING-08. 1097-1104.

Ying Zhang, Stephan Vogel, Alex Waibel. 2004. *Interpreting BLEU/NIST scores: How much improvement do we need to have a better system?* LREC-04

The corresponding authors of this paper are Hui Zhang (zhangh1982@gmail.com) and Min Zhang (mzhang@i2r.a-star.edu.sg)

Gazpacho and summer rash: lexical relationships from temporal patterns of web search queries

Enrique Alfonseca **Massimiliano Ciaramita** **Keith Hall**
Google
Zürich, Switzerland
ealfonseca@google.com, massi@google.com, kbhall@google.com

Abstract

In this paper we investigate temporal patterns of web search queries. We carry out several evaluations to analyze the properties of temporal profiles of queries, revealing promising semantic and pragmatic relationships between words. We focus on two applications: query suggestion and query categorization. The former shows a potential for time-series similarity measures to identify specific semantic relatedness between words, which results in state-of-the-art performance in query suggestion while providing complementary information to more traditional distributional similarity measures. The query categorization evaluation suggests that the temporal profile alone is not a strong indicator of broad topical categories.

1 Introduction

The temporal patterns of word occurrences in human communication carry an implicit measure of their relationship to real-world events and behavioral patterns. For example, when there is an event affecting a given entity (such as a natural disaster in a country), the entity name will turn up more frequently in human conversation, newswire articles and web documents; and people will search for it more often. Two entities that are closely related in the real world, such as the name of a country and a prominent region inside the country are likely to share common events and therefore be closely associated in human communication. Finally, two instances of the same class are also likely to share common usage patterns. For example, names of airlines or retail stores are more likely to be used by day rather than by night (Chien, 2005).

In this paper we explore the linguistic relationship between phrases that are judged to be similar based on their frequency time series correlation in search query logs. For every *phrase*[1] available in WordNet 3.0[2] (Miller, 1995), we have obtained its temporal signature from query logs, and calculated all their pairwise correlations. Next, we study the relationship in the top-ranked pairs with respect to their distribution in WordNet and a human-annotated labelling.

We also discuss possible applications of this data to solve open problems and present the results of two experiments: one where time series correlations turned out to be highly discriminative; and another where they were not particularly informative but shed some light on the nature of temporal semantics and topical categorization:

- *Query suggestion*, i.e. given a query, generate a ranked list of alternative queries in which the user may be interested.
- *Query categorization*, i.e. given a predefined set of categories, find the top categories to which the query can be assigned.

Finally, we illustrate with an example another application of time series in solving information extraction problems.

Although query logs are typically proprietary data, there are ongoing initiatives, like the Lemur toolbar[3], which make this kind of information available for research purposes. Other work (Bansal and Koudas, 2007b; Bansal and Koudas, 2007a) shows that temporal information can also be extracted from public data, such as blogs. More traditional types of text, such as news, are also typically associated with temporal labels; e.g., dates and timestamps.

This paper is structured in the following way:

[1] We use the term *phrase* to refer to any single word or multi-word expression that belongs to a synset in WordNet. Examples of phrases are *person*, *causal entity* or *william shakespeare*. We focused on the nouns hierarchy only.

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

[3] http://www.lemurproject.org/querylogtoolbar/

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1046–1055,
Singapore, 6-7 August 2009.

Section 2 summarizes the related work. Section 3 describes the correlation analysis between all pairs of phrases from WordNet. Next, Section 4 describes the application to query suggestion, and Section 5 the application to labelling queries in topical categories. Section 7 summarizes the conclusions and outlines ideas for future research.

2 Related work

The study of query time series explores a particular instance of the so-called *wisdom of the crowds* effect. Within this area, we can distinguish two kinds of phenomena. Knowledge and resources assembled by people explicitly, either individually, such as the case of blogs, or in a collaborative way, as in forums or wikis. These resources are valuable for human-consumption and can also be exploited in order to learn computational resources (Medelyan et al., 2008; Weld et al., 2008; Zesch et al., 2008b; Zesch et al., 2008a). On the other hand, it is possible to acquire useful resources and knowledge from aggregating behavioral patterns of large groups of people, even in the absence of a conscious effort. There is extensive ongoing research on the use of web search usage patterns to develop knowledge resources. Some examples are clustering co-click patterns to learn semantically related queries (Beeferman and Berger, 2000), combining co-click patterns with hitting times (Mei et al., 2008), analyzing query revisions made by users when querying search engines (Jones et al., 2006), replacing query words with other words that have the highest pointwise mutual information (Terra and Clarke, 2004), or using the temporal distribution of words in documents to improve ranking of search results (Jones and Diaz, 2007).

Within this second category, an important area is dedicated to the study of time-related features of search queries. News aggregators use real-time frequencies of user queries to detect spikes and identify news shortly after the spikes occur (Murata, 2008). Web users' query patterns have also proved useful for building a real-time surveillance system that accurately estimates region-by-region influenza activity with a lag of one day (Ginsberg et al., 2009). Search engines specifically developed for real-time searches, like Twitter search, will most likely provide new use cases and scenarios for quickly detecting trends in user search query patterns.

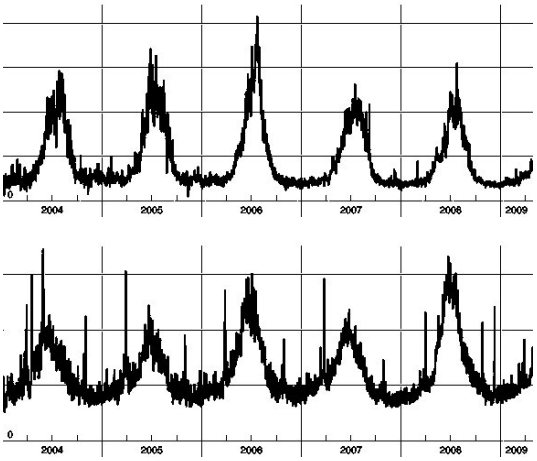


Figure 1: Time series obtained for the queries [gazpacho] and [summertime] (normalized scales).

Our study builds upon the work of Chien (2005), who observed that queries with highly-correlated temporal usage patterns are typically semantically related, and described a procedure for calculating the correlations efficiently. We have extended the analysis described in this work, by performing a more extensive evaluation of the kinds of semantic relationships that we can find among temporally-similar queries. We also propose, to our knowledge for the first time, areas of applications in solving well-established problems which shed some light on the nature of time-based semantic similarity. This work is also related to the analysis of temporal properties of information streams in data mining (Kleinberg, 2006) and information retrieval from time series databases (Agrawal et al., 1993).

3 Time-based similarities between phrases

Similarly to the method described in Chien (2005), we take a time interval, divide it into equally spaced subintervals, and represent each phrase of interest as the sequence of frequencies with which the phrase was observed in the subintervals. In our experiments, we have used as source data the set of fully anonymized query logs from the Google search engine between January 1st, 2004 and March 1st, 2009.[4].

These data have been aggregated on a daily basis so that we have the daily frequency of the

[4]Part of this data is publicly available from `http://www.google.com/trends`

queries of interest for over five years. The frequencies are then normalized with the total number of queries that happened on that day. The normalization is necessary to avoid daily and seasonal variations as there are typically more queries on weekdays than on weekends and fewer queries during holiday seasons than in the rest of the year. It also helps reducing the effect deriving from the fact that the population with Internet access is still monotonically growing, so we can expect that the number of queries will become higher and higher over time.

Given two phrases and their associated time series, the similarity metric used is the correlation coefficient between the two series (Chien, 2005). For illustration, Figure 1 shows the time series obtained for two sample queries, *gazpacho* and *summertime*, whose time series yield a correlation of 0.92. Similar high correlations can be observed with other queries related to phenomena that occur mainly in summer in the countries from which most queries come, like *summer rash*.

3.1 WordNet-based evaluation

In this section, we describe a study carried out with the purpose of discovering the traditional lexico-semantic relationships which hold between the queries that are most strongly related according to their temporal profiles.

For this evaluation, we have taken the nominal phrases appearing in WordNet 3.0. Given that users, when writing queries, typically do not pay attention to punctuation and case, we have normalized all phrases by lowercasing them and removing all punctuation. Next, we collected the time series for each phrase by computing the normalized daily frequency of each of them as exact queries in the query logs. The computation of the pairwise correlations was performed in parallel using the MapReduce infrastructure running over 2048 cores with 500 MB of RAM each. The total execution (including data shuffling and networking time) took approximately three hours.

Next, we represented the data as a complete graph where phrases are nodes and the edge between each pair of nodes is weighted by their time series correlation. Using a simple graph-cut we obtained clusters of related terms. A minimum weight threshold equal to 0.9 was applied;[5] thus, two phrases belong to the same cluster if there is a path between them only via edges with weight over 0.9.

The previous procedure produced a set of 604 clusters, with highly different sizes. The first observation is that 70% of the phrases in WordNet do not have a correlation over 0.9 with any other phrase, so they are placed alone in singleton clusters. There are several reasons for this. The clusters obtained are very specific: only phrases that have a very strong temporal association have temporal correlations exceeding the threshold. This is combined with the fact that we are using a very restricted vocabulary, namely the terms included in WordNet, which is many orders of magnitude smaller than the vocabulary of all possible queries from the users. Few phrase pairs in WordNet have a temporal association and popularity strong enough to be clustered together. Finally, many of the phrases in WordNet are rare, including scientific names of animals and plants, genuses or families, which are not commonly used. Therefore, the clusters extracted here correspond to very salient sets of phrases. If, instead of WordNet, we choose a vocabulary from known user queries (cf. Section 4), there would be many fewer singleton clusters, as the options of similar phrases to choose from would be much larger.

From the phrases that belong to clusters, 25% of the WordNet phrases do not have strong daily temporal profiles. The typical pattern for these terms is an almost flat time series, usually with small drops at summertime and Christmas (when seasonal leisure-related queries dominate). Therefore, these phrases were collected in just one cluster containing them all. Typical examples of the elements of this set are names of famous scientists and mathematicians (Gauss, Isaac Newton, Albert Einstein, Thomas Alva Edison, Hipprocrates, Gregor Mendel, ...), common terms (fertilization, famine, macroeconomics, genus, nationalism, ...), numbers and common first names, among other things. It is possible that using sub-day intervals might help to discriminate within this cluster.

The items in this big cluster contrast with periodical events, which display recurring patterns (e.g., queries related to elections or tax-returns), and names of famous people and other entities which appeared in the news in the past few years. All of these are associated with irregular, spiky time series. These constitute the final 5% of the

[5] This threshold is the same used by Chien (2005), and was confirmed after a manual inspection of a sample of the data

Type	Pairs	Examples
Synonyms	283	(angel cake, angel food cake), (thames, river thames), (armistice day, Nov 11)
Hyponym/hyperonyms	86	(howard hughes, aviator), (muhammad, prophet), (olga korbut, gymnast)
Siblings in hyponym taxonomy	611	(hiroshima, nagasaki), (junior school, primary school), (aids, welt)
Meronym/holonyms	53	(tutsi, rwanda), (july 4, july), (pyongyang, north korea)
Siblings in meronymy taxonomy	7	(everglades, everglades national park), (mississipi, orleans)
Other paths	471	(maundy thursday, maundy money), (tap water, water tap), (gren party, liberal)
Not structurally related	1009	(poppy, veterans day), (olympic games, gimnast), (belmont park, horse racing)

Table 1: Relationships between pairs of WordNet phrases belonging to the same cluster.

phrases belonging to small, highly focused, clusters.

Table 1 shows the relationships that hold between all pairs of phrases belonging to any of the smaller clusters. Out of 2520 pairs, 283 belong to the same synset, 697 are related via hyponymy links, 60 via meronymy links, and 471 by alternating hyponymy and meronymy links in the path. When the phrases were polysemous, the shortest path between any of their meaning was used. About 40% of the relations do not have a clear structural interpretation in WordNet.

The majority of pairs are related via more or less complex paths in the WordNet graph. Interestingly, even the structurally unrelated terms are characterized by transparent relations in terms of world knowledge, as it is the case between *poppy* and *veteran day*. Note as well that sometimes a WordNet term is used with a meaning not present in WordNet or in a different language, which may explain why *aids* has a very high correlation with *welt* (AIDS and welt are both hyponyms of *health problem*, but the correlation may be explained better by the AIDS World Day, *Welt Aids Tag* in German), and it also has a very high correlation with *sida*, defined in WordNet as a genus of tropical herbs, but which is in fact the translation of AIDS into Spanish. These observations motivated an additional manual labelling of the extracted pairs.

3.2 Hand labelled evaluation

As can be seen in Table 2, most of the terms that constitute a cluster are related to each other, although the kinds of semantic relationships that hold between them can vary significantly. Examples of the following kinds can be observed:

- **True synonyms**, as in the case of *november* and *nov*, or *architeuthis* and *giant squid*.
- **Variations of people names**, especially if a person's first name or surname is typically used to refer to that person, as in the case of *john lennon* and *lennon*, or *janis joplin* and *joplin*. Sometimes the variations include personal titles, as it is the case of *president carter* and *president nixon*, which are highly correlated with *jimmy carter* and *richard nixon*.
- **Geographically-related terms**, referring to locations which are located close to each other, as in the clusters {*korea, north korean, south korea, pyongyang, north korea*} and {*strasbourg, grenoble, toulouse, poitiers, lyon, lille, nantes, reims*}.
- **Synonyms of location names**, like *bahrain* and *bahrein*.
- **Derived words**, like *north korea* and *north korean*, or *lebanese* and *lebanon*.
- **Generic word optionalizations**, which happen when one word in a multi-word phrase is very correlated to the phrase, as in the case of *spanish inquisition* and *inquisition*, or *red bone marrow* and *red marrow*, where the most common interpretation for the shortened version of the phrase is the same as for the long version.
- **Word reordering**, where the two related phrases have the same words in a different order, as in the case of *maple sugar* and *sugar maple*, or *oil palm* and *palm oil*.
- **Morphological variants**: WordNet does not contain many morphological variants in the main dataset, but there are a few, like *station of the cross* and *stations of the cross*.
- **Acronyms**, like *federal emergency management agency* and *fema*.
- **Hyperonym-hyponym**, like *fern* and *plant*.
- **Sibling terms** in a taxonomy, as in the cluster {*lutheran, methodist, presbyterian, united methodist church, lutheran church,methodist church, presbyterian church,baptist, baptist church*}, which contains mostly names of Christian denominations.
- **Co-occurring events in time**, as is the case of *hitch* and *pacifier*, both titles of movies which were launched at almost the same

hydrant,fire hydrant
inauguration day,inauguration,swearing,investiture,inaugural address,inaugural,benediction,oath
indulgence,self indulgence
insulation,heating
interstate highway,interstate, intestine,small intestine
iq,iq test
irish people,irish,irish potato,irish gaelic,gaelic,irish soda bread,irish stew,st patrick,saint patrick,leprechaun, march 17,irish whiskey,shillelagh
ironsides,old ironsides
james,joyce,james joyce
janis joplin,joplin
jesus christ,pilate,pontius pilate,passion of christ,passion,aramaic
jewish new year,rosh hashana,rosh hashanah,shofar
john lennon,lennon
julep,mint julep,kentucky derby,kentucky
keynote,keynote address
kickoff,time off
korea,north korean,south korea,pyongyang,north korea
l ron hubbard,scientology
leap,leap year,leap day,february 29
left brain,right brain
leftover,leftovers,turkey stew
linseed oil,linseed
listeria,listeriosis,maple leaf
lobster tail,lobster,tails
lohan,lindsay
loire,rhone,rhone alpes
looking,looking for
lutheran,methodist,presbyterian,united methodist church,lutheran church,methodist church,presbyterian church, baptist,baptist church
mahatma gandhi,mahatma
malignant hyperthermia,hyperthermia
maple sugar,sugar maple
martin luther,martin luther king,luther,martin,martin luther king day
matzo,matzah,matzoh,passover,seder,matzo meal,pesach,haggadah,gefilte fish
mestizo,half blood,half and half
meteorology,weather bureau
moslem,muslim,prophet,mohammed,mohammad,muhammad,mahomet
movie star,star,revenge,film star,menace,george lucas
mt st helens,mount saint helens,mount st helens
myeloma,multiple myeloma
ness,loch ness,loch ness monster,loch,nessie
new guinea,papua new guinea,papua
november,nov
pacifier,hitch
papa,pope,vatican,vatican city,karol wojtyla,john paul ii,holy see,pius xii,papacy,paul vi,john xxiii,the holy see, vatican ii,pontiff,gulp,pater,nostradamus,ii,pontifex
parietal lobe,glioma,malignant tumor
particle accelerator,atom smasher,hadron,large,tallulah bankhead,bankhead,tanner
pledge,allegiance
president carter,jimmy carter
president nixon,richard nixon,richard m nixon
sept 11,september 11,sep 11,twin towers,wtc,ground zero,world trade center
slum,millionaire,pinto
strasbourg,grenoble,toulouse,poitiers,lyon,lille,nantes,reims
valentine,valentine day,february 14,romantic
aeon,flux
alien,predator
anne hathaway,hathaway
architeuthis,giant squid
basal temperature,basal body temperature
execution,saddam hussein,hussein,saddam,hanging,husain
flood,flooding
george herbert walker bush,george walker bush
intifada,palestine
may 1,may day,maypole

Table 2: Sample of clusters obtained from the temporal correlations.

Type	Clusters
True synonyms	19
Variations of people names	42
People names with and without titles	4
First name and surname from the same person	4
Geographically-related terms	18
Synonyms of location names	4
Derived words	4
Word optionalizations	87
Word reordering	7
Morphological variants	1
Acronyms	1
Cross-language synonyms	3
Hyperonym/hyponym	10
Sibling terms	10
Co-ocurring events in time	8
Topically related	38
Unrelated	72

Table 3: Results of the manual annotation of 2-item clusters.

time. A particular example of this is when the two terms are part of a named entity, as in the case of *quantum* and *solace*, which have a similar correlation because they appear together in a movie title.

- **Topically-related terms**, as the cluster {*jesus christ, pilate, pontius pilate, passion of christ, passion, aramaic*}, or the cluster containing popes and the Vatican. A similar example, *execution* is highly correlated to *saddam hussein*, because his execution attracted more interest worldwide during this time period than any other execution. Interestingly, topical correlation emerges at very specific granularity.

For the manual analysis of the results, we randomly selected 332 clusters containing only two items (so that 664 phrases were considered in total). Each of these pairs has been classified in one of the previous categories. The results of this analysis are shown in Table 3.

4 Application to query suggestion

Query suggestion is a feature of search engines that helps users reformulate queries in order to better describe their information need with the purpose of reducing the time needed to find the desired information (Beeferman and Berger, 2000; Kraft and Zien, 2004; Sahami and Heilman, 2006; Cucerzan and White, 2007; Yih and Meek, 2008). In this section, we explore the application of a similarity metric based on time series correlations for finding related queries to suggest to the users.

As a test set, we have used the query suggestion dataset from (Alfonseca et al., 2009). It contains a set of 57 queries and an average of 22 candidate query suggestions for each of them. Each suggestion was rated by two human raters using the 5-point Likert scale defined in (Sahami and Heilman, 2006), from irrelevant to highly relevant. The task involves providing a ranking of the suggestions that most closely resembles the human scores. The evaluation is based on standard IR metrics: precision at 1, 3 and 5, and mean average precision. In order to compute the precision- and recall-based metrics, we infer a binary distinction from the ratings: related or not related. The inter-annotator agreement for this dataset given the binary classification as computed by Cohen's Kappa is 0.6171.

Method	P@1	P@3	P@5	mAP
Random	0.37	0.37	0.37	0.43
Web Kernel	0.51	0.47	0.42	0.51
Dist. simil.	0.72	0.63	0.60	0.64
Time series	0.74	0.63	0.53	0.67
Combination	0.79	0.68	0.60	0.69

Table 4: Results for the query suggestion task.

We used three baselines: the average values that would be produced by a random scorer of the candidate suggestions, Sahami and Heilman (2006)'s system (based on calculating similarities between the retrieved snippets), and a recent competitive ranker based on calculating standard distributional similarities (Alfonseca et al., 2009) between the original query and the suggestion. Please refer to the referenced work for details.

In order to produce the ranked lists of candidate suggestions for each query, due to the lack of training data, we have opted for the unsupervised procedure described in the previous section:

1. Collect the daily time series of each of the queries and the candidate suggestions.
2. Calculate the correlation between the original query and each of the candidate suggestions provided for it, and use it as the candidate's score.
3. For each query, rank its candidate suggestions in decreasing order of correlation.

Finally, taking into account that the source of similarity is very different to the one used for distributional similarity, we tested the hypothesis that

a combination of the two techniques would be beneficial to capture different features of the queries and suggestions. We have trained a linear mixture model combining both scores (time series and distributional similarities), using 10-fold cross validation.

The results are displayed in Table 4. For evaluating the results, whenever a system produced a tie between several suggestions, we generated 100 random orderings of the elements in the tie, and report the average scores.

Using distributional similarities and the temporal series turned out to be indistinguishable for the precision scores at 0.95 confidence, and both are significantly better than the similarity metric based on the web kernel. The combination produced an improvement across all metrics, although not statistically significant at p=0.05.

This is quite a positive finding as the time series method relies on stored information requiring only simple and highly optimized lookups.

5 Application to query categorization

The results from the manual evaluation in Section 3.2 support the conclusion that time series from query logs provide powerful signals for clustering at a fine-grained level, in some cases uncovering synonyms (may 1st, may day) and even causal relations (insulation, heating). A natural question is if temporal information is correlated with other types of categorizations. In this section we carry out a preliminary exploration of the relation between query time series and query categorization. To this extent we adapt the data from the KDD 2005 CUP (Li et al., 2005), which provides a set of queries classified into 67 broad topical categories. Since the data is rather sparse (678 queries) we applied Fourier analysis to "smooth" the time series.

5.1 The KDD CUP data

The KDD Cup 2005 [6] introduced a query categorization task and dataset consisting of 800,000 unlabeled queries for unsupervised training, and an evaluation set of 911 queries, 111 for development and 800 for the final evaluation. The systems submitted for this task can be quite complex and made full use of the large unlabeled set. Our goal here is not to provide a comparative evaluation, but only

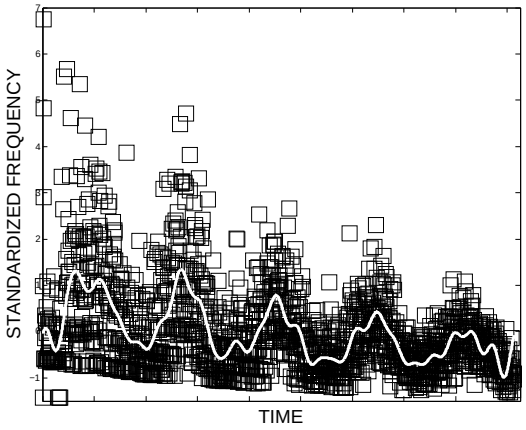


Figure 2: RDFT reconstruction for the query "brush cutters" using the first 25 Fourier coefficients. The squares represent the original time series datapoints, while the continuous line represents the reconstructed signal.

to use the labelled data[7] in a simplified manner to better understand the semantic properties of query time series. Each query in the dataset is assessed by three editors who can assign multiple topic labels from a set of 67 categories belonging to seven broad topics: Computers, Entertainment, Information, Living, Online Community, Shopping and Sports. We merged the KDD Cup development and test set, out of the 911 queries we were able to retrieve significant temporal information for 678 queries. We joined the sets of labels from each assessor for each query. On average, each query is assigned five labels.

5.2 DFT analysis

Assessing the similarity of data represented as time series has been addressed mostly my means of Fourier analysis; e.g., Agrawal et al. (1993) introduce a method for efficiently retrieving time series from databases based on Discrete Fourier Transform (DFT). Several other methods have been proposed, e.g., Discrete Wavelet Transform (DWT), however DFT provide a competitive benchmark approach (Wu et al., 2000).

We use DFT to generate the Fourier coefficients of the time series and Reverse DFT (RDFT) to reconstruct the original signal using only a subset of the coefficients. This analysis effectively compresses the time series producing a smoother approximate representation. DFT can be computed efficiently via Fast Fourier Transform (FFT), with

[6] http://www.sigkdd.org/kdd2005/kddcup.html

[7] The KDD Cup dataset is probably the only public query log providing topical categorization information.

Method	Accuracy	± std-err
Random	0.107	0.03
MostFrequent	0.490	0.07
DFT-c10	0.425	0.06
DFT-c50	0.456	0.05
DFT-c100	0.502	0.05
DFT-c200	0.456	0.04
DFT-c400	0.506	0.05
DFT-c600	0.481	0.06
DFT-c800	0.478	0.04
DFT-c1000	0.466	0.05

Table 5: Results of the KDD dataset exploration.

complexity $O(n \log n)$ where n is the length of the sequence. The approximate representation is useful not only to address sparsity but can also be used to efficiently estimate the similarity of two time series using only a small subset of coefficients as in (Agrawal et al., 1993). As an example, Figure 2 shows the original time series for the query "brush cutters" and its reconstructed signal using only the first 25 Fourier coefficients. The reconstructed signal captures the essence of the periodicity of the query and highlights the yearly peaks registered for the query in spring and summer.

5.3 Experiment and discussion

To explore the correlation between the structured temporal representation of queries provided by the time series and topical categorization we run the following experiment. Each KDD Cup query was reconstructed via RDFT using a variable number of coefficients. The set of 679 queries was partitioned in 10 sets and a 10-fold evaluation was performed. For each fold we trained a classifier on the remaining 9 folds. We used an average multi-class perceptron (Freund and Schapire, 1999) adapted to multi-label learning (Crammer and Singer, 2003). Each model was trained on a fixed number of 10 iterations. The accuracy of each model was evaluated as the fraction of test items for which the selected highest scoring class was in the gold standard set provided by the editors. As a lower bound we estimated the accuracy of randomly choosing a label for each test instance, and as a baseline we used the most frequent label. The latter is a powerful predictor: baselines based on class frequency outperform most of the systems that participated in the KDD Cup (Lin and Wu, 2009).

Table 5 reports the average accuracy over the 10 runs with relative standard errors. Each DFT-based model is characterized by the number of coefficients used for the reconstruction. Two main patterns are noticeable. First, none of the differences between the frequency-based baseline and the DFT models is significant, this seems to indicate that temporal structure alone is not a good discriminator of topic, at least of broad categories. In retrospect, this is somewhat predictable. The temporal dimension is a basic semantic component of lexical meaning and world knowledge which is not necessarily associated with any broad, and to some extent subjective, categorization. An inspection of the patterns found in each category shows in fact that similar patterns often emerge in different categories; e.g., "Halloween costume" and "cheesecake recipe" have a similar yearly periodical pattern with spikes in early winter, while monotonically decaying patterns are shared across all categories; e.g., between computer hardware and kids toys.

The second interesting finding is the trend of the DFT system results, higher at low-intermediate values, providing some initial promising evidence that DFT analysis generates useful compressed representations which could be indexed and applied efficiently. Notice that the sequences reconstructed using 1,000 coefficients reproduce almost identically the original signals.

6 Applications in information extraction

Time series from query logs are particularly relevant for phrases that refer to entities which are involved in recent events. Therefore, we expect them to be useful for solving other applications that require handling entities, such as named entity recognition and classification, relation extraction or disambiguation.

To illustrate this point, we mention an example of relation extraction between actors and movies: movies usually have spikes when they are released, and then the frequency again drops sharply. At the same times, when a movie is released, the search engine users have a renewed interest in their actors. Figure 3 displays the time series for the five most recent movies by Jim Carrey (as of march 2009), and the time series for Jim Carrey. As can be seen, the spikes are at exactly the same points in time. If we add up the series (a) through (e) into a single series and calculate the correlation with (f), it turns out to be very high (0.88).

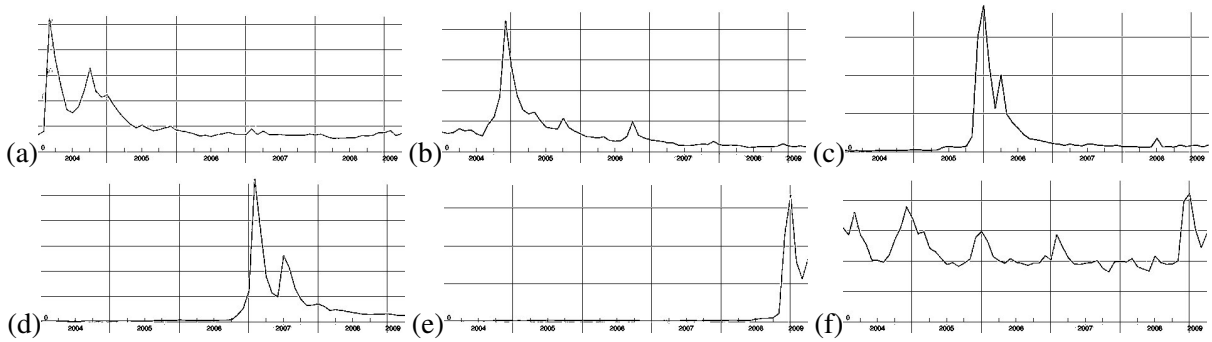


Figure 3: Time series obtained for the five most recent movies with Jim Carrey, and (f) time serie for the query [jim carrey] (normalized scales).

System	Precision	Recall	F-measure
Random	0.24	0.14	0.17
Time series	0.53	0.66	0.57

Table 6: Results for the query suggestion task.

To validate the hypothesis that this data should be useful for identifying related entities, we have performed a small experiment in the following way: by choosing five popular actors[8] and the cinema movies in which they appear since the year 2004, obtained from IMDB[9]. Using the time series, for each actor we choose the combination of movies such that, by adding up the time series of those movies, we maximise the correlation with the actor's time series. It has been implemented with a greedy beam search, with a beam size of 100. The results are shown in Table 6. The random baseline randomly associates the movies from the dataset with the five actors.

We do not believe this to be a perfect feature as, for example, actors may have a peak in the time series related to their personal lives, not necessarily to movies. However, the high correlations that can be obtained when the pairing between actors and movies is correct, and the improvement with respect a random baseline, indicates this is a feature which can probably be integrated with other relation extraction systems when handling relationships between entities that have big temporal dependencies.

7 Conclusions and future work

This paper explores the relationships between queries whose associated time series obtained from query logs are highly correlated. The use of time series in semantic similarity has been discussed by Chien (2005), but only a very preliminary evaluation was described, and, to our knowledge, they had never been applied and evaluated in solving existing problems. Our results indicate that, for a substantial percentage of phrases in a thesaurus, it is possible to find other highly-related phrases; and we have categorized the kind of semantic relationships that hold between them.

We have found that in a query suggestion task, somewhat surprisingly, results are comparable with other state-of-the-art techniques based on distributional similarities. Furthermore, information obtained from time series seems to be complementary with them, as a simple combination of similarity metrics produces an important increase in performance..

From an analysis on a query categorization task the initial evidence suggests that there is no strong correlation between broad topics and temporal profiles. This agrees with the intuition that time provides a fundamental semantic dimension possibly orthogonal to broad topical classification. This issue however deserves further investigation. Another issue which is worth a deeper investigation is the application of Fourier transform methods which offer tools for studying the periodic structure of the temporal sequences.

[8] Ben Stiller, Edward Norton, Jim Carrey, Leonardo Dicaprio, and Tom Hanks.

[9] www.imdb.com.

References

R. Agrawal, C. Faloutsos, and A.N. Swami. 1993. Efficient similarity search in sequence databases. In *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, pages 69–84.

E. Alfonseca, K. Hall, and S. Hartmann. 2009. Large-scale computation of distributional similarities for queries. In *Proceedings of North American Chapter of the Association for Computational Linguistics - Human Language Technologies conference*.

N. Bansal and N. Koudas. 2007a. BlogScope: a system for online analysis of high volume text streams. In *Proceedings of the 33rd international conference on Very large data bases*, pages 1410–1413.

N. Bansal and N. Koudas. 2007b. BlogScope: Spatio-temporal analysis of the blogosphere. In *Proceedings of the 16th international conference on World Wide Web*, pages 1269–1270.

D. Beeferman and A. Berger. 2000. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416.

S. Chien. 2005. Semantic similarity between search engine queries using temporal correlation. In *Proceedings of the 14th international conference on World Wide Web*, pages 2–11.

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

S. Cucerzan and R.W. White. 2007. Query suggestion based on user landing pages. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 875–876.

Y. Freund and R.E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37:277–296.

J. Ginsberg, M.H. Mohebbi, R.S. Patel, L. Brammer, M.S. Smolinski, and L. Brilliant. 2009. Detecting influenza epidemics using search engine query data. *Nature*, 457, February.

R. Jones and F. Diaz. 2007. Temporal profiles of queries. *ACM Transactions on Information Systems*, 25(3):14.

R. Jones, B. Rey, O. Madani, and W. Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, pages 387–396.

J. Kleinberg. 2006. Temporal dynamics of on-line information streams. In *Data Stream Management: Processing High-Speed Data*. Springer.

R. Kraft and J. Zien. 2004. Mining anchor text for query refinement. In *Proceedings of the 13th international conference on World Wide Web*, pages 666–674.

Y. Li, Z. Zheng, and H. Dai. 2005. KDD Cup-2005 report: Facing a grat challenge. *SIGKDD Explor. Newsl.*, 7(2):91–99.

D. Lin and X. Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*.

O. Medelyan, C. Legg, D. Milne, and I.H. Witten. 2008. *Mining meaning from Wikipedia*. Dept. of Computer Science, University of Waikato.

Q. Mei, D. Zhou, and K. Church. 2008. Query suggestion using hitting time. In *Proceeding of the 17th ACM conference on Information and knowledge management*, pages 469–478.

G.A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

T. Murata. 2008. Detection of breaking news from online web search queries. *New Generation Computing*, 26(1):63–73.

M. Sahami and T.D. Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web*, pages 377–386.

E. Terra and C.L.A. Clarke. 2004. Scoring missing terms in information retrieval tasks. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 50–58.

D.S. Weld, F. Wu, E. Adar, S. Amershi, J. Fogarty, R. Hoffmann, K. Patel, and M. Skinner. 2008. Intelligence in Wikipedia. In *Proceedings of the 23rd Conference on Artificial Intelligence*.

Y. Wu, D. Agrawal, and A. El Abbadi. 2000. A comparison of DFT and DWT based similarity search in time-series databases. In *Proceedings of the 9th International ACM Conference on Information and Knowledge Management*, pages 488–495.

W. Yih and C. Meek. 2008. Consistent Phrase Relevance Measures. *Workshop on Data Mining and Audience Intelligence for Advertising*, page 37.

T. Zesch, C. Muller, and I. Gurevych. 2008a. Extracting lexical semantic knowledge from Wikipedia and Wiktionary. In *Proceedings of the Conference on Language Resources and Evaluation*.

T. Zesch, C. Muller, and I. Gurevych. 2008b. Using Wiktionary for computing semantic relatedness. In *Proceedings of the Conference on Artificial Intelligence*, pages 861–867.

A Compact Forest for Scalable Inference over Entailment and Paraphrase Rules

Roy Bar-Haim[§], **Jonathan Berant**[⋆], **Ido Dagan**[§]

[§]Computer Science Department, Bar-Ilan University, Ramat Gan 52900, Israel
{barhair,dagan}@cs.biu.ac.il

[⋆]The Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel
jonatha6@post.tau.ac.il

Abstract

A large body of recent research has been investigating the acquisition and application of applied inference knowledge. Such knowledge may be typically captured as *entailment rules*, applied over syntactic representations. Efficient inference with such knowledge then becomes a fundamental problem. Starting out from a formalism for entailment-rule application we present a novel packed data-structure and a corresponding algorithm for its scalable implementation. We proved the validity of the new algorithm and established its efficiency analytically and empirically.

1 Introduction

Applied semantic inference is concerned with deriving target meanings from texts. In the textual entailment framework, this is reduced to inferring a textual statement (the *hypothesis* h) from a source *text* (t). Traditional formal semantics approaches perform such inferences over logical forms derived from the text. By contrast, most practical NLP applications operate over shallower representations such as parse trees, possibly supplemented with limited semantic information about named entities, semantic roles etc.

Most commonly, inference over such representations is made by applying some kind of transformations or substitutions to the tree or graph representing the text. Such transformations may be generally viewed as *entailment (inference) rules*, which capture semantic knowledge about paraphrases, lexical relations such as synonyms and hyponyms, syntactic variations etc. Such knowledge is either composed manually, e.g. WordNet (Fellbaum, 1998), or learned automatically.

A large body of work has been dedicated to learning paraphrases and entailment rules, e.g. (Lin and Pantel, 2001; Shinyama et al., 2002; Szpektor et al., 2004; Bhagat and Ravichandran, 2008), identifying appropriate contexts for their application (Pantel et al., 2007) and utilizing them for inference (de Salvo Braz et al., 2005; Bar-Haim et al., 2007). Although current available rule bases are still quite noisy and incomplete, the progress made in recent years suggests that they may become increasingly valuable for text understanding applications. Overall, applied knowledge-based inference is a prominent line of research gaining much interest, with recent examples including the series of workshops on *Knowledge and Reasoning for Answering Questions (KRAQ)*[1] and the planned evaluation of knowledge resources in the forthcoming 5^{th} Recognizing Textual Entailment challenge (RTE-5)[2].

While many applied systems utilize semantic knowledge via such inference rules, their use is typically limited, application-specific, and quite heuristic. Formalizing these practices seems important for applied semantic inference research, analogously to the role of well-formalized models in parsing and machine translation. Bar-Haim et al. (2007) made a step in this direction by introducing a generic formalism for semantic inference over parse trees. Their formalism uses entailment rules as a unifying representation for various types of inference knowledge, allowing unified inference as well. In this formalism, rule application has a clear, intuitive interpretation as generating a new sentence parse (a *consequent*), semantically entailed by the source sentence. The inferred consequent may be subject to further rule applications

[1] http://www.irit.fr/recherches/ILPL/kraq09.html
[2] http://www.nist.gov/tac/2009/RTE/

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1056–1065,
Singapore, 6-7 August 2009.

and so on. In their implementation, each consequent was generated explicitly as a separate tree.

Following this line of work, our long-term research goal is to investigate effective utilization of entailment rules for inference. While the formalism of Bar-Haim et al. provides a principled framework for modeling such inferences, its implementation using explicit generation of consequents raises severe efficiency issues, since the number of consequents may grow exponentially in the number of rule applications. Consider, for example, the sentence *"Children are fond of candies."*, and the following entailment rules: *'children→kids'*, *'candies→sweets'*, and *'X is fond of Y→X likes Y'*. The number of derivable sentences (including the source sentence) would be 2^3 as each rule can either be applied or not, independently. Indeed, we found that this exponential explosion leads to poor scalability in practice. Intuitively, we would like that each rule application would add just the entailed part (e.g. *kids*) to a packed sentence representation. Yet, we still want the resulting structure to represent a set of entailed sentences, rather than some mixture of sentence fragments whose semantics is unclear.

As discussed in section 5, previous work proposed only partial solutions to this problem. In this paper we present a novel data structure, termed *compact forest*, and a corresponding inference algorithm, which efficiently generate and represent all consequents while preserving the identity of each individual one (section 3). Our work is inspired by previous work on packed representations in other fields, such as parsing, generation and machine translation (section 5). As we follow a well-defined formalism, we could prove that all inference operations of Bar-Haim et al. are equivalently applied over the compact forest. We compare inference cost over compact forests to explicit consequent generation both theoretically (section 3.4), illustrating an exponential-to-linear complexity ratio, and empirically (section 4), showing improvement by orders of magnitude. These results suggest that our data-structure and algorithm are both valid and scalable, opening up the possibility to investigate large-scale entailment rule application within a well-formalized framework.

2 Inference Framework

This section briefly presents a (simplified) description of the tree transformations inference formalism of Bar-Haim et al. (2007). Given a source text, syntactically parsed, and a set of *entailment rules* representing tree transformations, the formalism defines the set of consequents derivable from the text using the rules. Each consequent is obtained through a sequence of rule applications, each generates an intermediate parse tree, similar to a proof process in logic.

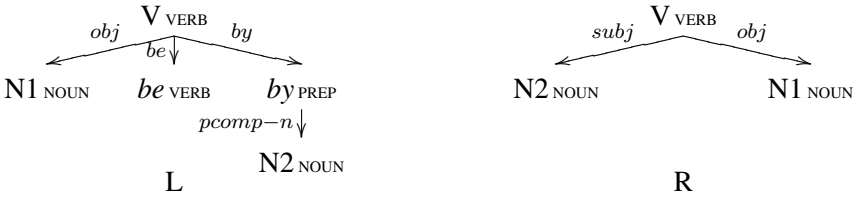
More specifically, sentences are represented as dependency trees, where nodes are annotated with lemma and part-of-speech, and edges are annotated with dependency relation. A rule '$L \rightarrow R$' is primarily composed of two *templates*, termed *left-hand-side* (L), and *right-hand-side* (R). Templates are dependency subtrees which may contain POS-tagged *variables*, matching any lemma. Figure 1(a) shows passive-to-active transformation rule, and (b) illustrates its application.

A rule application generates a derived tree d from a source tree s through the following steps:

L matching: First, a match of L in the source tree s is sought. In our example, the variable V is matched in the verb *see*, $N1$ is matched in *Mary* and $N2$ is matched in *John*.

R instantiation: Next, a copy of R is generated and its variables are instantiated according to their matching node in L. In addition, a rule may specify *alignments*, defined as a partial function from L nodes to R nodes. An alignment indicates that for each modifier m of the source node that is not part of the rule structure, the subtree rooted at m should also be copied as a modifier of the target node. In addition to defining alignments explicitly, each variable in L is implicitly aligned to its counterpart in R. In our example, the alignment between the V nodes implies that *yesterday* (modifying *see*) should be copied to the generated sentence, and similarly *beautiful* (modifying *Mary*) is copied for $N1$.

Derived tree generation: Let r be the instantiated R, along with its descendants copied from L through alignment, and l be the subtree matched by L. The formalism has two methods for generating the derived tree d: *substitution* and *introduction*, as specified by the rule type. *Substitution* rules specify modification of a subtree of s, leaving the rest of s unchanged. Thus, d is formed by copying s while replacing l (and the descendants



(a) Passive to active transformation (substitution rule)

Source: Beautiful Mary was seen by John yesterday.

Derived: John saw beautiful Mary yesterday.

(b) Application of passive to active transformation

Figure 1: An inference rule application. POS and relation labels are based on Minipar (Lin, 1998)

of l's nodes) with r. This is the case for the passive rule, as well as for lexical rules such as '*buy* → *purchase*'. By contrast, *introduction* rules are used to make inferences from a subtree of s, while the other parts of s are ignored and do not affect d. A typical example is inferring a proposition embedded as a relative clause in s. In this case, the derived tree d is simply taken to be r.

In addition to inference rules, the formalism includes *annotation rules* which add features to existing parse tree nodes. These rules have been used for identifying contexts that affect the polarity of predicates.

As shown by Bar-Haim et al., this concise, well defined formalism allows unified representation of diverse types of knowledge which are commonly used for applied semantic inference.

3 Efficient Inference over Compact Parse Forests

As shown in the introduction, explicit generation of consequents (henceforth *explicit inference*) leads to an exponential explosion of the number of generated trees. In this section we present our efficient implementation for this formalism. Our implementation is based on a novel data structure, termed *compact forest* (Section 3.1), which compactly represents a large set of trees. Each rule application generates explicitly only the nodes of the rule right-hand-side while the rest of the consequent tree is shared with the source, which also reduces the number of redundant rule applications. As we shall see, this novel representation is based primarily on *disjunction edges*, an extension of dependency edges that specify a set of alternative edges of multiple trees. Section 3.2 presents an efficient algorithm for inference over compact forests, followed by a discussion of its correctness and complexity (sections 3.3 and 3.4).

3.1 The Compact Forest Data Structure

A compact forest $\mathcal{F}$ represents a set of dependency trees. Figure 2 shows an example of a compact forest, containing both the source and derived sentences of Figure 1. We first define a more general data structure for directed graphs, and then narrow the definition to the case of trees.

A *Compact Directed Graph (cDG)* is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of **disjunction edges** *(d-edges)*. Let $\mathcal{D}$ be a set of dependency relations. A d-edge d is a triple (S_d, rel_d, T_d), where S_d and T_d are disjoint sets of source nodes and target nodes; $rel_d : S_d \rightarrow \mathcal{D}$ is a function specifying the dependency relation corresponding to each source node. Graphically, d-edges are shown as point nodes, with incoming edges from source nodes and outgoing edges to target nodes. For instance, let d be the bottom-most d-edge in Figure 3. Then $S_d = \{$*of, like*$\}$, $T_d = \{$*candy, sweet*$\}$, $rel(of) =$ *pcomp-n*, and $rel($*like*$) =$ *obj*.

A d-edge represents, for each $s_i \in S_d$, a set of

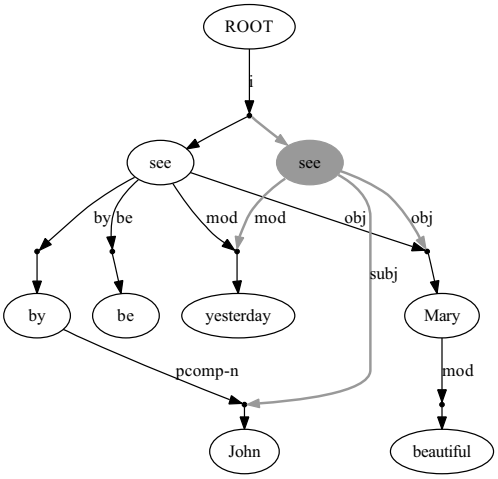


Figure 2: A compact forest containing both the source and derived sentences of Figure 1. Parts of speech are omitted.

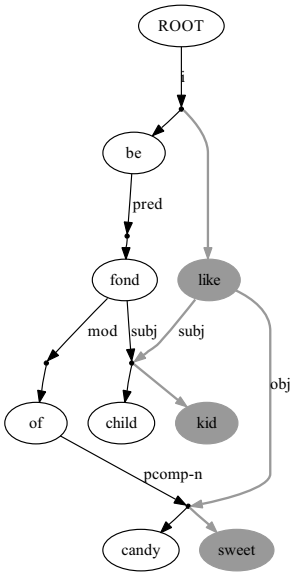


Figure 3: A compact forest representing the 2^3 sentences derivable from the sentence *"children are fond of candies"* using the following three rules: *'children→kids'*, *'candies→sweets'*, and *'X is fond of Y→X likes Y'*.

alternative directed edges $\{(s_i, t_j) : t_j \in T_d\}$, all of which are labeled with the same relation given by $rel_d(s_i)$. Each of these edges, termed *embedded edge (e-edge)*, would correspond to a different graph represented in $\mathcal{G}$. In the previous example, the e-edges are *like* $\xrightarrow{obj}$ *candy*, *like* $\xrightarrow{obj}$ *sweet*, *of* $\xrightarrow{pcomp-n}$ *candy* and *of* $\xrightarrow{pcomp-n}$ *sweet* (notice that the definition implies that all source nodes in S_d have the same set of alternative target nodes T_d). d is called an *outgoing d-edge* of a node v if $v \in S_d$ and an *incoming d-edge* of v if $v \in T_d$. A *Compact Directed Acyclic Graph (cDAG)* is a cDG that contains no cycles of e-edges.

A DAG G rooted in a node $v \in \mathcal{V}$ of a cDAG $\mathcal{G}$ is *embedded* in $\mathcal{G}$ if it can be derived as follows: we initialize G with v alone; then, we *expand* v by choosing exactly one target node $t \in T_d$ from each outgoing d-edge d of v, and adding t and the corresponding e-edge (v, t) to G. This expansion process is repeated recursively for each new node added to G.

Each such set of choices results in a different DAG with v as its only root. In Figure 2, we may choose to connect the root either to the left *see*, resulting in the source passive sentence, or to the right *see*, resulting in the derived active sentence.

A **Compact Forest** $\mathcal{F}$ is a cDAG with a single root r (i.e. r has no incoming d-edges) where all the embedded DAGs rooted in r are trees. This set of trees, termed *embedded trees*, comprise the set of trees represented by $\mathcal{F}$.

Figure 3 shows another example for a compact forest efficiently representing the 2^3 sentences resulting from three independently-applied rules.

3.2 The Inference Process

Next, we describe the algorithm implementing the inference process of Section 2 over the compact forest (henceforth, *compact inference*), illustrating it through Figures 1 and 2.

Forest initialization $\mathcal{F}$ is initialized with the set of dependency trees representing the *text* sentences, with their roots connected under the forest root as the target nodes of a single d-edge. Dependency edges are transformed trivially to d-edges with a single source and target. Annotation rules are applied at this stage to the initial $\mathcal{F}$. The black part of Figure 2 corresponds to the initial forest.

Rule application comprises the following steps:

L matching: L is matched in $\mathcal{F}$ if there exists an embedded tree t in $\mathcal{F}$ such that L is matched in t, as in Section 2. We denote by l the subtree of t in which L was matched. As in section 2, the match in our example is $(V, N1, N2)$=*(see, Mary, John)*. Notice that this definition does not allow l to be scattered over multiple embedded trees.

As the target nodes of a d-edge specify alternatives for the same position in the tree, their parts-of-speech are expected to be of substitutable types.

In this paper we further assume that all target nodes of the same d-edge have the same part-of-speech[3]. Consequently, variables that are leaves in L and may match a certain target node of a d-edge d are mapped to the whole set of target nodes T_d rather than to a single node. This yields a compact representation of multiple matches, and prevents redundant rule applications. For instance, given a compact representation of '{*Children/kids*} *are fond of* {*candies/sweeets*}' (cf. Figure 3), the rule *'X is fond of Y→X likes Y'* will be matched and applied only once, rather than four times (for each combination of matching X and Y).

Derived tree generation: A template r consisting of R while excluding variables that are leaves of both L and R (termed *dual leaf-variables*)[4] is generated and inserted into $\mathcal{F}$. In case of a substitution rule (as in our example), r is set as an alternative to l by adding r's root to T_d, where d is the incoming d-edge of l's root. In case of an introduction rule, it is set as an alternative to the other trees in the forest by adding r's root to the target node set of the forest root's outgoing d-edge. In our example, r is the gray node (still labeled with the variable V) , and it becomes an additional target node of the d-edge entering the original (left) *see*.

Variable instantiation: Each variable in r (i.e. a non-dual leaf) is instantiated according to its match in L (as in Section 2), e.g. V is instantiated with *see*. As specified above, if the variable is a leaf in L is not a dual leaf then it is matched in a set of nodes, and hence each of them should be instantiated in r. This is decomposed into a sequence of simpler operations: first, r is instantiated with a representative from the set, and then we apply (ad-hoc) lexical substitution rules for creating a new node for each other node in the set[5].

Alignment sharing: Modifiers of aligned nodes are shared (rather than copied) as follows. Given a node n_L in l aligned to a node n_R in r, and an outgoing d-edge d of n_L which is not part of l, we share d between n_L and n_R by adding n_R to S_d and setting $rel_d(n_R) = rel_d(n_L)$. This is illustrated by the sharing of *yesterday* in Figure 2. We also copy annotation features from n_L to n_R.

We note at this point that the instantiation of variables that are not dual leaves (e.g. V in our example) cannot be shared because they typically have different modifiers at the two sides of the rule. Yet, their modifiers which are not part of the rule are shared through the alignment operation (recall that common variables are always considered aligned). Dual leaf variables, on the other hand, might be shared, as described next, since the rule doesn't specify any modifiers for them.

Dual leaf variable sharing: This final step is performed analogously to alignment sharing. Suppose that a dual leaf variable X is matched in a node v in l whose incoming d-edge is d. Then we simply add the parent p of X in r to S_d and set $rel_d(p)$ to the relation between p and X (in R). Since v itself is shared, its modifiers become shared as well, implicitly implementing the alignment operation. The subtrees *beautiful Mary* and *John* are shared this way for variables $N1$ and $N2$.

Applying the rule in our example added only a single node and linked it to four d-edges, compared to duplicating the whole tree in explicit inference.

3.3 Correctness

In this section we present two theorems, which prove that the inference algorithm is a valid implementation of the inference formalism of Section 2. Due to space limitations, the proofs themselves are omitted, and instead we outline their general scheme.

We first argue that performing any sequence of rule applications over the set of initial trees results in a compact forest:

Theorem 1: *The compact inference process generates a compact forest.*

Proof scheme: We prove by induction on the number of rule applications. Initialization generates a single-rooted cDAG, whose embedded DAGs are all trees, as required. We then prove that if applying a rule on a compact forest creates a cycle or an embedded DAG that is not a tree, then such a cycle or a non-tree DAG already existed prior to rule application, in contradiction with the inductive assumption. A crucial observation for this proof is that for any path from a node u to a node v that passes through r, where u and v are

[3]This is the case in our current implementation, which is based on the coarse tag-set of Minipar (Lin, 1998).

[4]With the following exceptions: variables that are the only node in R (and hence are both the root and a leaf), and variables with additional alignments (other than the implicit alignment between their occurrences in L and R) are not considered dual-leaves.

[5]Notice that these nodes, in addition to the usual alignment with their source nodes in l, share the same daughters in r.

outside r, there is also an analogous path from u to v that passes through l instead, QED.

Next, we argue that the inference process over a compact forest is complete and sound, i.e., it generates the set of consequents derivable from a text according to the inference formalism.

Theorem 2: *Given a rule base $\mathcal{R}$ and a set of initial trees T, a tree t is embedded in a compact forest derivable from T by the compact inference process $\Leftrightarrow$ t is a consequent of T according to the inference formalism.*

Proof scheme: We first show completeness by induction on the number of explicit rule applications. Let t_{n+1} be a tree derived from a tree t_n using the rule r_n according to the inference formalism. The inductive assumption asserts that t_n is embedded in some derivable compact forest $\mathcal{F}$. It is easy to verify that applying r_n to $\mathcal{F}$ will yield a compact forest $\mathcal{F}'$ in which t_{n+1} is embedded.

Next, we show soundness by induction on the number of rule applications over the compact forest. Let t_{n+1} be a tree represented in some derived compact forest $\mathcal{F}_{n+1}$. $\mathcal{F}_{n+1}$ was derived from the compact forest $\mathcal{F}_n$, using the rule r_n. It can be shown that $\mathcal{F}_n$ represents a tree t_n, such that applying r_n on t_n will yield t_{n+1} according to the formalism. The inductive assumption asserts that t_n is a consequent in the inference formalism and therefore t_{n+1} is a consequent as well, QED.

These two theorems guarantee that the compact inference process is valid - i.e., it yields a compact forest that represents the set of consequents derivable from a given text by a given rule set.

3.4 Complexity

In this section we explain why compact inference exponentially reduces the time and space complexity in typical scenarios.

We consider a set of rule matches in a tree T *independent* if their matched left-hand-sides (excluding dual-leaf variables) do not overlap in T, and their application over T can be chained in any order. For example, the three rule matches presented in Figure 3 are independent.

Let us consider explicit inference first. Assume we start with a single tree T with k independent rules matched. Applying k rules will yield 2^k trees, since any subset of the rules might be applied to T. Therefore, the time and space complexity of applying k independent rule matches is $\Omega(2^k)$. Applying more rules on the newly derived consequents behaves in a similar manner.

	Compact	Explicit	Ratio
Time (msec)	61	24,184	396
Rule applications	12	123	10
Node count	69	5,901	86
Edge endpoints	141	11,552	82

Table 1: Compact vs. explicit inference, using generic rules. Results are averaged per text-hypothesis pair.

Next, we examine compact inference. Applying a rule using compact inference adds the right-hand-side of the rule and shares with it existing d-edges. Since that the size of the right-hand-side and the number of outgoing d-edges per node are practically bounded by low constants, applying k rules on a tree T yields a linear increase in the size of the forest. Thus, the resulting size is $O(|T|+k)$, as we can see from Figure 3.

The time complexity of rule application is composed of matching the rule in the forest and applying the matched rule. Applying a matched rule is linear in its size. Matching a rule of size r in a forest $\mathcal{F}$ takes $O(|\mathcal{F}|^r)$ time even when performing an exhaustive search for matches in the forest. Since r tends to be quite small and can be bounded by a low constant, this already gives polynomial time complexity. In practice, indexing the forest nodes, as well as the typical low connectivity of the forest, result in a very fast matching procedure, as illustrated in the empirical evaluation, described next.

4 Empirical Evaluation

This section reports empirical evaluation of the efficiency of compact inference, tested in the recognizing textual entailment setting using the RTE-3 and RTE-4 datasets (Giampiccolo et al., 2007; Giampiccolo et al., 2009). These datasets consist of *(text, hypothesis)* pairs, which need to be classified as *entailing/non entailing*. Our first experiment shows, using a small rule set, that compact inference outperforms explicit inference by orders of magnitude (Section 4.1). The second experiment shows that compact inference scales well to a full-blown RTE setting with several large-scale rule bases, where up to hundreds of rules are applied for a text (Section 4.2).

4.1 Compact vs. Explicit Inference

To compare explicit and compact inference we randomly sampled 100 pairs from the RTE-3 development set, and parsed the *text* in each pair using Minipar (Lin, 1998). We used a set of manually-composed entailment rules for inference over generic linguistic phenomena such as passive, conjunction, relative clause, apposition, possessives, and determiners, which contains a few dozens of rules. To make a fair comparison, we aimed to make the explicit inference implementation reasonably efficient, e.g. by preventing generation of the same tree by different permutations of the same rule applications. Both configurations perform rule application iteratively, until no new matches are found. In each iteration we first find all rule matches and then apply all matching rules. We compare run time, number of rule applications, and the overall generated size of nodes and edges, where edge size is represented by the sum of its endpoints.

The results are summarized in Table 1. As expected, the results show that compact inference is by orders of magnitude more efficient than explicit inference. To avoid memory overflow, inference was terminated after reaching 100,000 nodes. 3 out of the 100 pairs reached that limit with explicit inference, while the maximal node count for compact inference was only 268. The number of rule applications is reduced thanks to the sharing of common subtrees in the compact forest, by which a single rule application operates simultaneously over a large number of embedded trees. The results suggest that scaling to larger rule bases and longer inference chains would be feasible for compact inference, but prohibitive for explicit inference.

4.2 Application to an RTE System

Experimental setting The goal of the second experiment was to assess that compact inference scales well for broad entailment rule bases. In this experiment we used the Bar-Ilan RTE system (Bar-Haim et al., 2009). The system operates in two primary stages: ***Inference***, in which entailment rules are applied to the initial compact forest $\mathcal{F}$, aiming to bring it closer to the hypothesis $\mathcal{H}$, and ***Classification***, in which a set of features is extracted from the resulting $\mathcal{F}$ and from $\mathcal{H}$ and fed into an SVM classifier, which determines entailment.

The classification setting and its features are quite typical for the RTE literature. They include lexical and structural measures for the coverage of $\mathcal{H}$ by $\mathcal{F}$, where high coverage is assumed to correlate with entailment, as well as features aiming to detect inconsistencies between $\mathcal{F}$ and $\mathcal{H}$ such as incompatible arguments for the same predicate or incompatible verb polarity (see below). For a complete feature description, see (Bar-Haim et al., 2009).

Rule Bases In addition to the *generic rules* described in Section 4.1, the following large-scale sources for entailment rules were used: ***Wikipeda:*** We used the lexical rulebase of Shnarch et al. (2009), who extracted rules such as *'Janis Joplin → singer'* from Wikipedia based on both its metadata (e.g. links and redirects) and text definitions, using patterns such as '*X is a Y*'. ***WordNet:*** We extracted from WordNet (Fellbaum, 1998) lexical rules based on synonyms, hypernyms and derivation relations. ***DIRT:*** The DIRT algorithm (Lin and Pantel, 2001) learns from a corpus entailment rules between binary predicates, e.g. *'X is fond of Y→X likes Y'*. We used the version described in (Szpektor and Dagan, 2007), which learns canonical rule forms. ***Argument-Mapped WordNet (AmWN):*** A resource for entailment rules between verbal and nominal predicates (Szpektor and Dagan, 2009), including their argument mapping, based on WordNet and NomLexplus (Meyers et al., 2004), verified statistically through intersection with the unary-DIRT algorithm (Szpektor and Dagan, 2008). In total, these rule bases represent millions of rules. ***Polarity Annotation Rules:*** We compiled a small set of annotation rules for marking the polarity of predicates as *negative* or *unknown* due to verbal negation, modal verbs, conditionals etc. (Bar-Haim et al., 2009).

Search In this work we focus on efficient representation of the search space, leaving for future work the complementary problem of devising effective search heuristics over our representation. In the current experiment we implemented a simple search strategy, in the spirit of (de Salvo Braz et al., 2005): first, we applied three exhaustive iterations of generic rules. Since these rules have low fan-out (few possible right-hand-sides for a given left-hand-side) it is affordable to apply and chain them more freely. We then perform a single iteration of all other lexical and lexical-syntactic rules,

applying them only if their L part was matched in $\mathcal{F}$ and their R part was matched in $\mathcal{H}$.

The system was trained over the RTE-3 development set, and tested on both RTE-3 test set and RTE-4 (which includes only a test set).

Results Table 2 provides statistics on rule application using all rule bases, over the RTE-3 development set and the RTE-4 dataset[6]. Overall, the primary result is that the compact forest indeed accommodates well extensive rule application from large-scale rule bases. The resulting forest size is kept small, even in the maximal cases which were causing memory overflow for explicit inference.

The accuracies obtained in this experiment and the overall contribution of rule-based inference are shown in Table 3. The results on RTE-3 are quite competitive: compared to our 66.4%, only 3 teams out of the 26 who participated in RTE-3 scored higher than 67%, and three more systems scored between 66% and 67%. The results for RTE4 rank 9-10 out of 26, with only 6 teams scoring higher by more than 1%. Overall, these results validate that the setting of our experiment represents a state-of-the-art system.

Inference over the rule bases utilized in our experiment improved the accuracy on both test sets. The contribution was more prominent for the RTE-4 dataset. These results illustrate a typical contribution of current knowledge sources for current RTE systems. This contribution is likely to increase with current and near future research, on topics such as extending and improving knowledge resources, applying them only in semantically suitable contexts, improved classification features and broader search strategies. As for our current experiment, we may conclude that the goal of assessing the compact forest scalability in a state-of-the-art setting was achieved [7].

Finally, Tables 4 and 5 illustrate the usage and contribution of individual rule bases. Table 4 shows the distribution of rule applications over the various rule bases. Table 5 presents ablation study showing the marginal accuracy gain for each rule base. These results show that each of the rule bases is applicable for a large portion of the pairs, and contributes to the overall accuracy.

[6]Running time is omitted since most of it was dedicated to rule fetching, which was rather slow for our available implementation of some resources. The elapsed time was a few seconds per (t, h) pair.

[7]We note that common RTE research issues, such as improving accuracy, fall out of the scope of the current paper.

	RTE3-Dev		RTE4	
	Avg.	Max.	Avg.	Max.
Rule applications	14	275	15	110
Node count	71	606	80	357
Edge endpoints	155	1,741	173	1,062

Table 2: Application of compact inference to the RTE-3 Dev. and RTE-4 datasets, using all rule types.

	Accuracy		
Test set	No inference	Inference	Δ
RTE3	64.6%	66.4%	1.8%
RTE4	57.5%	60.6%	*3.1%

Table 3: Inference contribution to RTE performance. The system was trained on the RTE-3 development set. * indicates statistically significant difference (at level $p < 0.02$, using McNemar's test).

Rule base	RTE3-Dev		RTE4	
	Rules	App	Rules	App
WordNet	0.6	1.2	0.6	1.1
AmWN	0.3	0.4	0.3	0.4
Wikipedia	0.6	1.7	0.6	1.3
DIRT	0.5	0.7	0.5	1.0
Generic	4.7	10.4	5.4	11.5
Polarity	0.2	0.2	0.2	0.2

Table 4: Average number of rule applications per (t, h) pair, for each rule base. *App* counts each rule application, while *Rules* ignores multiple matches of the same rule in the same iteration.

Rule base	ΔAccuracy (RTE4)
WordNet	0.8%
AmWN	0.7%
Wikipedia	1.0%
DIRT	0.9%
Generic	0.4%
Polarity	0.9%

Table 5: Contribution of various rule bases. Results show accuracy loss on RTE-4, obtained for removing each rule base (ablation tests).

5 Related Work

This section discusses related work on applying knowledge-based transformations within RTE systems, as well as on using packed representations in other NLP tasks.

RTE Systems Previous RTE systems usually restricted both the type of allowed transformations and the search space. Systems based on lexical (word-based or phrase-based) matching of h in t typically applied only lexical rules (without vari-

ables), where both sides of the rule are matched directly in t and h (Haghighi et al., 2005; MacCartney et al., 2008). The inference formalism we use is more expressive, allowing also syntactic and lexical-syntactic transformations as well as rule chaining.

Hickl (2008) derived from a given (t, h) pair a small set of *discourse commitments*, which are quite similar to the kind of consequents we derive by our syntactic and lexical-syntactic rules. The commitments were generated by several different tools and techniques, compared to our generic unified inference process, and commitment generation efficiency was not discussed.

Braz et al. (2005) presented a semantic inference framework which "augments" the text representation with only the right-hand-side of an applied rule, and in this respect is similar to ours. However, in their work, both rule application and the semantics of the resulting "augmented" structure were not fully specified. In particular, the distinction between individual consequents was lost in the augmented graph. By contrast, our compact inference is fully formalized and is provably equivalent to an expressive, well-defined formalism operating over individual trees, and each inferred consequent can be recovered from the compact forest.

Packed representations Packed representations in various NLP tasks share common principles, which also underly our compact forest: factoring out common substructures and representing choice as local disjunctions. Applying this general scheme to individual problems typically requires specific representations and algorithms, depending on the type of alternatives that should be represented and the specified operations for creating them. In our work, alternatives are created by rule application, where a newly derived subtree is set as an alternative to existing subtrees. Alternatives are specified locally using d-edges.

Packed chart representations for parse forests were introduced in classical parsing algorithms such as CYK and Earley (Jurafsky and Martin, 2008), and have been extended in later work for various purposes (Maxwell III and Kaplan, 1991; Kay, 1996). Alternatives in the parse chart stem from syntactic ambiguities, and are specified locally as the possible decompositions of each phrase into its sub-phrases.

Packed representations have been utilized also in transfer-based machine translation. Emele and Dorna (1998) translated packed source language representation to packed target language representation while avoiding unnecessary unpacking during transfer. Unlike our rule application, in their work transfer rules *preserve* ambiguity stemming from source language, rather than *generating* new alternatives. Mi et al.(2008) applied statistical machine translation to a source language parse forest, rather than to the 1-best parse. Their transfer rules are tree-to-string, contrary to our tree-to-tree rules, and chaining is not attempted (rules are applied in a single top-down pass over the source forest), and thus their representation and algorithms are quite different from ours.

6 Conclusion

This work addresses the efficiency of entailment and paraphrase rule application. We presented a novel compact data structure and a rule application algorithm for it, which are provably a valid implementation of a generic inference formalism. We illustrated inference efficiency both analytically and empirically. Beyond entailment inference, we suggest that the compact forest would also be useful in generation tasks (e.g. paraphrasing). Our efficient representation of the consequent search space opens the way to future investigation of the benefit of larger-scale rule chaining, and to the development of efficient search strategies required to support such inferences.

Acknowledgments

This work was partially supported by the PASCAL-2 Network of Excellence of the European Community FP7-ICT-2007-1-216886, the FBK-irst/Bar-Ilan University collaboration and the Israel Science Foundation grant 1112/08. The second author is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship. The authors would like to thank Yonatan Aumann, Marco Pennacchiotti and Marc Dymetman for their valuable feedback on this work.

References

Roy Bar-Haim, Ido Dagan, Iddo Greental, and Eyal Shnarch. 2007. Semantic inference at the lexical-syntactic level. In *Proceedings of AAAI*.

Roy Bar-Haim, Jonathan Berant, Ido Dagan, Iddo Greental, Shachar Mirkin, Eyal Shnarch, and Idan

Szpektor. 2009. Efficient semantic deduction and approximate matching over compact parse forests. In *Proceedings of the First Text Analysis Conference (TAC 2008)*.

Rahul Bhagat and Deepak Ravichandran. 2008. Large scale acquisition of paraphrases for learning surface patterns. In *Proceedings of ACL-08: HLT*.

Rodrigo de Salvo Braz, Roxana Girju, Vasin Punyakanok, Dan Roth, and Mark Sammons. 2005. An inference model for semantic entailment in natural language. In *Proceedings of AAAI*.

Martin C. Emele and Michael Dorna. 1998. Ambiguity preserving machine translation using packed representations. In *Proceedings of Coling-ACL*.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech and Communication. MIT Press.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The Third PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Danilo Giampiccolo, Hoa Trang Dang, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2009. The Fourth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the First Text Analysis Conference (TAC 2008)*.

Aria D. Haghighi, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via graph matching. In *Proceedings of EMNLP*.

Andrew Hickl. 2008. Using discourse commitments to recognize textual entailment. In *Proceedings of COLING*.

Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, second edition.

Martin Kay. 1996. Chart generation. In *Proceedings of ACL*.

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

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Proceedings of the Workshop on Evaluation of Parsing Systems at LREC*.

Bill MacCartney, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP*.

John T. Maxwell III and Ronald M. Kaplan. 1991. A method for disjunctive constraint satisfaction. In Masaru Tomita, editor, *Current Issues in Parsing Technology*. Kluwer Academic Publishers.

A. Meyers, R. Reeves, Catherine Macleod, Rachel Szekeley, Veronkia Zielinska, and Brian Young. 2004. The cross-breeding of dictionaries. In *Proceedings of LREC*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL-08: HLT*.

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. 2007. ISP: Learning inferential selectional preferences. In *Proceedings of NAACL-HLT*.

Yusuke Shinyama, Satoshi Sekine, Kiyoshi Sudo, and Ralph Grishman. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of Human Language Technology Conference (HLT 2002)*, San Diego, USA.

Eyal Shnarch, Libby Barak, and Ido Dagan. 2009. Extracting lexical reference rules from Wikipedia. In *Proceedings of ACL-IJCNLP*.

Idan Szpektor and Ido Dagan. 2007. Learning canonical forms of entailment rules. In *Proceedings of RANLP*.

Idan Szpektor and Ido Dagan. 2008. Learning entailment rules for unary templates. In *Proceedings of COLING*.

Idan Szpektor and Ido Dagan. 2009. Augmenting WordNet-based inference with argument mapping. In *Proceedings of ACL-IJCNLP Workshop on Applied Textual Inference (TextInfer)*.

Idan Szpektor, Hristo Tanev, Ido Dagan, and Bonaventure Coppola. 2004. Scaling web based acquisition of entailment patterns. In *Proceedings of EMNLP*.

Discriminative Substring Decoding for Transliteration

Colin Cherry and **Hisami Suzuki**
Microsoft Research
One Microsoft Way
Redmond, WA, 98052
{colinc,hisamis}@microsoft.com

Abstract

We present a discriminative substring decoder for transliteration. This decoder extends recent approaches for discriminative character transduction by allowing for a list of known target-language words, an important resource for transliteration. Our approach improves upon Sherif and Kondrak's (2007b) state-of-the-art decoder, creating a 28.5% relative improvement in transliteration accuracy on a Japanese katakana-to-English task. We also conduct a controlled comparison of two feature paradigms for discriminative training: indicators and hybrid generative features. Surprisingly, the generative hybrid outperforms its purely discriminative counterpart, despite losing access to rich source-context features. Finally, we show that machine transliterations have a positive impact on machine translation quality, improving human judgments by 0.5 on a 4-point scale.

1 Introduction

Transliteration occurs when a word is borrowed into a language with a different character set. The word is transcribed into the new character set in such a way as to maintain rough phonetic correspondence; for example, the English word *hip-hop* becomes ヒップホップ [*hippuhoppu*], when transliterated into Japanese. A task frequently of interest to the NLP community is back-transliteration, where one seeks the original word, given the borrowed form.

We investigate machine transliteration as a method to handle out-of-vocabulary items in a Japanese-to-English translation system. More often than not, this will correspond to back-transliteration. Our goal is to prevent the copying or deletion of Japanese words when they are missing from our statistical machine translation (SMT) system's translation tables. This can have a substantial impact on the quality of SMT output, transforming translations of questionable usefulness, such as:

> Avoid using a フリーメール account.[1]

into the far more informative:

> Avoid using a Freemail account.

Though the techniques we present here are language-independent, we focus this study on the task of Japanese katakana-to-English back-transliteration. Katakana is one of the four character types used in the Japanese writing system (along with hiragana, kanji and Roman alphabet), consisting of about 50 syllabic characters. It is used primarily to spell foreign loanwords (e.g., チョコレート [*chokoreeto*] — *chocolate*), and names (e.g., クリントン [*kurinton*] — *Clinton*). Therefore, katakana is a strong indicator that a Japanese word can be back-transliterated. However, katakana can also be used to spell scientific names of animals and plants (e.g., カモ [*kamo*] — *duck*), onomatopoeic expressions (e.g., バシャバシャ [*bashabasha*] — *splash*) and foreign origin words that are not transliterations (e.g., ホチキス [*hochikisu*] — *stapler*). These untransliterable cases constitute about 10% of the katakana words in our data.

We employ a discriminative substring decoder for machine transliteration. Following Sherif and Kondrak (2007b), the decoder operates on short source substrings, with each operation producing one or more target characters, as shown in Figure 1. However, where previous approaches employ generative modeling, we use structured perceptron training to discriminatively tune parameters according to 0-1 transliteration accuracy. This

[1] フリーメール is romanized as [*furiimeeru*]

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1066–1075,
Singapore, 6-7 August 2009.

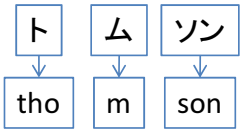


Figure 1: Example substring derivation

allows us to test novel methods for the use of target lexicons in discriminative character transduction, allowing our decoder to benefit from a list of known target words. Perhaps more significantly, our framework allows us to test two competing styles of features:

- sparse **indicators**, designed to capture the same channel and language modeling data collected by previous generative models, and
- components of existing generative models, used as real-valued features in a discriminatively weighted, **generative hybrid**.

Note that generative hybrids are the norm in SMT, where translation scores are provided by a discriminative combination of generative models (Och, 2003). Substring-based transliteration with a generative hybrid model is very similar to existing solutions for phrasal SMT (Koehn et al., 2003), operating on characters rather than words. Unlike out-of-the-box phrasal SMT solutions, our generative hybrid benefits from a target a lexicon. As we will show, this is the difference between a weak baseline and a strong competitor.

We demonstrate that despite recent successes in discriminative character transduction using indicator features (Jiampojamarn et al., 2008; Dreyer et al., 2008), our generative hybrid performs surprisingly well, producing our highest transliteration accuracies. Researchers frequently compare against a phrasal SMT baseline when evaluating a new transduction technique (Freitag and Khadivi, 2007; Dreyer et al., 2008); however, we are careful to vary only the features in our comparison. Confounding variables, such as alignment, decoder and training method, are held constant.

We also include a human evaluation of transliteration-augmented SMT output. Though human evaluations are too expensive to allow a comparison between transliteration systems, we are able to show that adding our transliterations to a production-level SMT engine results in a substantial improvement in translation quality.

2 Background

This work draws inspiration from previous work in transliteration, which we divide into similarity and transduction-based approaches. We also discuss recent successes in discriminative character transduction that have influenced this work.

2.1 Similarity-based transliteration

In similarity-based transliteration, a character-based, cross-lingual similarity metric is calculated (or bootstrapped) from known transliteration pairs. Given a source word s, its transliteration is the target word t most similar to s, where t is drawn from some pool of candidates. This approach may also be referred to as **transliteration discovery**.

Brill et al. (2001) describe a katakana-to-English approach with an EM-learned edit distance, which bootstraps from a small number of examples to learn transliteration pairs from query logs. Bilac and Tanaka (2005) harvest transliteration candidates from comparable bilingual corpora (conference abstracts in English and Japanese), and use distributional as well as phonetic similarity to choose among them. Sherif and Kondrak (2007a) also bootstrap a learned edit distance for Arabic named entities, with candidate pairs drawn from sentence or document-aligned parallel text. Klementiev and Roth (2006) bootstrap an SVM classifier trained to detect true transliteration-pairs. They draw candidates from comparable news text, using date information to provide further clues as to aligned named entities. Bergsma and Kondrak (2007) extend the classification approach with features derived from a character alignment. They train from bilingual dictionaries and word-aligned parallel text, selecting negative examples to target false-friends.

The work of Hermjakob et al. (2008) is particularly relevant to this paper, as they incorporate a similarity-based transliteration system into an Arabic-to-English SMT engine. They employ a hand-crafted cross-lingual similarity metric, and use capitalized n-grams from the Google n-gram corpus as candidates. With such a huge candidate list, a cross-lingual indexing scheme is designed for fast candidate look-up. Their work also addresses the question of when to transliterate (as opposed to translate), a realistic concern when deploying a transliteration component in SMT. This, however, is not of so much concern for katakana, as it is used primarily for loanwords.

2.2 Transduction-based transliteration

The approach presented in this paper is an instance of transduction-based transliteration, where the source word is transformed into a target word using a sequence of character-level operations. The parameters of the transduction process are learned from a collection of transliteration pairs. These systems do not require a list of candidates, but many incorporate a target lexicon, favoring target words that occur in the lexicon. This approach is also known as **transliteration generation**.

The majority of transliteration generation approaches are based on the noisy channel model, where a target t is generated according to $P(t|s) \propto P(s|t)P(t)$. This approach is typified by finite-state transliteration, where the various stages of the channel model are represented by finite state transducers and automata. Early systems employed a complex channel, passing through multiple phonetic representations (Knight and Graehl, 1998; Bilac and Tanaka, 2004), but later versions replaced characters directly (Al-Onaizan and Knight, 2002). Sherif and Kondrak (2007b) extend this approach with substring operations in the style of phrasal SMT, and show that doing so improves both accuracy as well as space and time efficiency. Note that it is possible to incorporate a target lexicon by making $P(t)$ a word unigram model with a character-based back-off.

Li et al. (2004) present an alternative to the noisy channel with their joint n-gram model, which calculates $P(s,t)$. This formulation allows operations to be conditioned on both source and target context. However, the inclusion of a candidate list is more difficult in this setting, as $P(t)$ is not given its own model.

Zelenko and Aone (2006) investigate a purely discriminative, alignment-free approach to transliteration generation. The target word is constructed one character at a time, with each new character triggering a suite of features, including indicators for near-by source and target characters, as well a generative target language model. Freitag and Khadivi (2007) propose a discriminative, latent edit distance for transliteration. In this case, training data need not be aligned in advance, but a latent alignment is produced during decoding. Again, the target word is constructed one character at a time, using edit operations that are scored according to source and target context features. Both approaches train using a structured perceptron, as we do here. However, these models represent a dramatic departure from the existing literature, while ours has clear analogs to the well-known noisy-channel paradigm, which allows for useful comparisons and insights into the advantages of discriminative training.

2.3 Discriminative character transduction

While our chosen application is transliteration, our decoder is influenced by recent successes in general-purpose discriminative transduction. Jiampojamarn et al. (2008) describe a discriminative letter-to-phoneme substring transducer, while Dreyer et al. (2008) describe a discriminative character transducer with a latent derivation structure for morphological transformations. Both models are extremely effective, but both rely exclusively on indicator features; they do not explore the use of knowledge-rich generative models. Our indicator system uses an extended version of the Jiampojamarn et al. (2008) feature set.

3 Methods

We adopt a discriminative substring decoder for our transliteration task. A structured perceptron (Collins, 2002) learns weights for our transliteration features, which are drawn from two broad classes: indicator and hybrid generative features.

3.1 Structured perceptron

The decoder's discriminative parameters are learned with structured perceptron training. Let a derivation d describe a substring operation sequence that transliterates a source word into a target word. Given an input training corpus of such derivations $D = d_1 \ldots d_n$, a vector feature function on derivations $\vec{F}(d)$, and an initial weight vector $\vec{w}$, the perceptron performs two steps for each training example $d_i \in D$:

- Decode: $\bar{d} = \text{argmax}_{d \in D(\text{src}(d_i))} \left(\vec{w} \cdot \vec{F}(d) \right)$
- Update: $\vec{w} = \vec{w} + \vec{F}(d_i) - \vec{F}(\bar{d})$

where $D(\text{src}(d))$ enumerates all possible derivations with the same source side as d. To improve generalization, the final feature vector is the average of all vectors found during learning (Collins, 2002). Accuracy on the development set is used to select the number of times we pass through all $d_i \in D$.

Given the above framework, we require training derivations D, feature vectors $\vec{F}$, and a decoder to

carry out the argmax over all d reachable from a particular source word. We describe each of these components in turn below.

3.2 Training derivations

Note that the above framework describes a max-derivation decoder trained on a corpus of gold-standard derivations, as opposed to a max-transliteration decoder trained directly on source-target pairs. By building the entire system on the derivation level, we side-step issues that can occur when perceptron training with hidden derivations (Liang et al., 2006), but we also introduce the need to transform our training source-target pairs into training derivations.

Training derivations can be learned unsupervised from source-target pairs using character alignment techniques. Previously, this has been done using an EM-learned edit distance (Ristad and Yianilos, 1998), or generalizations thereof (Brill and Moore, 2000; Jiampojamarn et al., 2007). We opt for an alternative alignment technique, similar to the word-aligner described by Zhang et al. (2008). This approach employs variational EM with sparse priors, along with hard length limits, to reduce the length of substrings operated upon. By doing so, we hope to learn only non-compositional transliteration units.

Our aligner produces only monotonic alignments, and does not allow either the source or target side of an operation to be empty. The same restrictions are imposed during decoding. In this way, each alignment found by variational EM is also an unambiguous derivation. We align our training corpus with a maximum substring length of three characters. The same derivations are used to train all of the transliteration systems tested in this paper.

3.3 Features

We employ two main types of features: indicators and hybrid generative models. Indicators detect binary events in a derivation, such as the presence of a particular operation. Hybrid generative features assign a real-valued probability to a derivation, based on statistics collected from training derivations. There are few generative features and each carries a substantial amount of information, while indicators are sparse and knowledge-poor.

We treat these two classes of features as distinct. We do so because researchers often use either one approach or the other.[2] Furthermore, it is not clear how to optimally employ training derivations when combining generative models and sparse indicators: generative models need large amounts of data to collect statistics and relatively little for perceptron training,[3] while sparse indicators require only a large perceptron training set.

We can further divide feature space according to the information required to calculate each feature. Both feature sets can be partitioned into the following subtypes:

- **Emission**: How accurate are the operations used by this derivation?
- **Transition**: Does the target string produced by this derivation look like a well-formed target character sequence?
- **Lexicon**: Does the target string contain known words from a target lexicon?

Indicator Features

Previous approaches to discriminative character transduction tend to employ only sparse indicators (Jiampojamarn et al., 2008; Dreyer et al., 2008). This is because sparsity is not a major concern in character-based domains, and sparse indicators are extremely flexible.

Our emission and transition indicator features follow Jiampojamarn et al. (2008). Emission indicators are centered around an operation, such as $[\vdash \rightarrow \textit{tho}]$. Minimally, an indicator exists for each operation. Many more **source context** features can be generated by conjoining an operation with source n-grams found within a fixed window of C characters to either side of the operation. These source context features have minimal computational cost, and they allow each operator to account for large, overlapping portions of the source, even when the substrings being operated upon are small. Meanwhile, transition indicators stand in for a character-based target language model. Indicators are built for each possible target n-gram, for $n = 1 \ldots K$, allowing the perceptron to construct a discriminative back-off model. Development experiments lead us to select $C = 3$ and $K = 5$.

[2] Generative hybrids are often accompanied by a small number of unsparse indicators, such as operation count.

[3] Perceptron training on the same data used for model construction can lead to overconfidence in model quality. One can address this problem by using a large number of modeling-training folds (Collins et al., 2005), but we do not do so here.

Indicator lexicon features are novel to this work. Given access to a target lexicon with type frequencies, we opt to create features that indicate the frequencies of generated target words according to coarse bins. Experiments on our development set lead to the selection of 5 frequency bins: [$<$ 2,000], [$<$ 200], [$<$ 20], [$<$ 2], [$<$ 1]. To keep the model linear, these features are cumulative; thus, generating a word with frequency 126 will result in both the $[< 2,000]$ and $[< 200]$ features firing. Note that a single transliteration can potentially generate multiple target words, and doing so can have a major impact on how often the lexicon features fire. Thus, we employ another feature that indicates the introduction of a new word. We expect these frequency indicators to be superior to a word-level unigram model, as they allow the designer to select notable frequencies. In particular, the bins we have selected do not give any advantage to extremely common words, as these are generally less likely to be transliterated.

Hybrid Generative Features

We begin with the three components of the generative noisy channel employed by Sherif and Kondrak (2007b). Their transliteration probability is:

$$P(t|s) \propto P_E(s|t) \cdot \max\left[P_T(t), P_L(t)\right] \quad (1)$$

Inspired by the linear models used in SMT (Och, 2003), we can discriminatively weight the components of this generative model, producing:

$$w_E \log P_E(s|t) + w_T \log P_T(t) + w_L \log P_L(t)$$

with weights w learned by perceptron training.

These three models conveniently align with our three feature subtypes. Emission information is provided by $P_E(s|t)$, which is estimated by maximum likelihood on the operations observed in our training derivations. Including source context is difficult in such a model. To compensate for this, all systems using $P_E(s|t)$ also use composed operations, which are constructed from operation sequences observed in the training set. This removes the length limit on substring operations.[4] $P_T(t)$ provides transition information through a character language model, estimated on the target side of the training derivations. In our implementation, we employ a KN-smoothed 7-gram model (Kneser and Ney, 1995). Finally, $P_L(t)$ is a unigram target word model, estimated from the same type frequencies used to build our lexicon indicators.

Since we have adopted a linear model, we are no longer constrained by the original generative story. Therefore, we are free to incorporate other SMT-inspired features: $P_{E'}(t|s)$, target character count, and operation count.[5]

Feature summary

The indicator and hybrid-generative feature sets each provide a discriminative version of the noisy channel model. In the case of transition and lexicon features, both systems have access to the exact same information, but encode that information differently. The lexicon encoding is the most dramatic difference, with the indicators using a small number of frequency bins, and the generative unigram model providing a single, real-valued feature that is proportional to frequency.

In the case of their emission features, the two systems actually encode different information. Both have access to the same training derivations, but the indicator system provides source context through n-gram indicators, while the generative system does so using composed operations.

3.4 Decoder

Our decoder builds upon machine translation's monotone phrasal decoding (Zens and Ney, 2004), or equivalently, the sequence tagging algorithm used in semi-Markov CRFs (Sarawagi and Cohen, 2004). This dynamic programming (DP) decoder extends the Viterbi algorithm for HMMs by operating on one or more source characters (a substring) at each step. A DP block stores the best scoring solution for a particular prefix. Each block is subdivided into cells, which maintain the context necessary to calculate target-side features. We employ a beam, keeping only the 40 highest-scoring cells for each block, which speeds up inference at the expense of optimality. We found that the beam had no major effect on perceptron training, nor on the system's final accuracy.

Previously, target lexicons have been used primarily in finite-state transliteration, as they are easily encoded as finite-state-acceptors (Al-Onaizan and Knight, 2002; Sherif and Kondrak,

[4] Derivations built by our character aligner use operations on substrings of maximum length 3. To enable perceptron training with composed operations, once $P_E(s|t)$ has been estimated by counting composed operations in the initial alignments, we re-align our training examples with those composed operations to maximize $P_E(s|t)$, creating new training derivations.

[5] Character and operation counts also fit in the indicator system, but did not improve performance in development.

2007b). It is possible to extend the DP decoder to also use a target lexicon. By encoding the lexicon as a trie, and adding the trie index to the context tracked by the DP cells, we can provide access to frequency estimates for words and word prefixes. This has the side-effect of creating a new cell for each target prefix; however, in the character domain, this remains computationally tractable.

4 Data

4.1 Wikipedia training and test data

Our katakana-to-English training data is derived from bilingually-linked Wikipedia titles. Any Japanese Wikipedia article with an entirely katakana title and a linked English article results in training pair. This results in 60K transliteration pairs; we removed 2K pairs for development, and 2K for held-out testing.

The remaining 56K training pairs are quite noisy. As mentioned earlier, roughly 10% of our examples are simply not transliterable, but approximate Wikipedia title translations are an even more substantial source of noise. For example, コンピュータゲーム [*konpyuutageemu*] — *computer game* is aligned with the English article *Computer and video games*. We found it beneficial, in terms of both speed and accuracy, to do some coarse alignment-based pruning. After alignment, the operations used by all derivations are counted. Any operation that is used fewer than three times is eliminated, along with any derivation using that operation. The goal is to eliminate loose transliteration pairs from our data, where a word or initial is included in one language but not the other. This results in 40K training pairs. Despite the noise in the Wikipedia data, there are clear advantages in using it for training transliteration models: it is available for any language pair, it reflects recent trends and events, and the amount of data increases daily. As we will see below, the model trained on this data performs well on a test set from a very different domain.

All systems use development set accuracy to select their meta-parameters, such as the number of perceptron iterations, the size of the source-context window, and the n-gram length used in character language modeling. The hybrid generative system further splits the training set, using 38K derivations for the calculation of its emission and transition models, and 2K derivations for perceptron training its model weights.

4.2 Machine translation test data

In order to see how effective our transliterator is on out-of-domain test data, we also created test data from a log of translation requests to a web-based, Japanese-to-English translation service.[6] Out of 5,000 randomly selected translation requests, there are 312 cases where katakana source words are out-of-vocabulary for the MT system, and therefore remain untranslated. We created a reference translation (not necessarily a transliteration) for these katakana words by manually selecting the corresponding English word(s) in the sentence-level reference translation, which was produced independently from this experiment. This test set is quite divergent from the Wikipedia titles: only 17 (5.5%) of its katakana words are found in the Wikipedia training data, and six of these did not agree on the English translation.

4.3 English lexicon

Our English lexicon is derived from two overlapping data sources: the English gigaword corpus (LDC2003T05; GW) and the language model training data for our SMT system, which contains selections from Europarl, gigaword, and web-harvested text. Both are lowercased. We combine the unigram frequency counts from the two sources by taking the max when they overlap. The resulting lexicon has 5M types, 2.5M of which have frequency 1.

5 Experiments

In this section, we summarize development experiments, and then conduct a comparison on our two transliteration test sets. We report 0-1 accuracy: a transliteration is only correct if it exactly matches the reference. For the comparison experiments, we also report 10-best accuracy, where a system is correct if it includes the correct transliteration somewhere in its 10-best list.

5.1 Baselines

We compare our systems against a re-implementation of Sherif and Kondrak's (2007b) noisy-channel substring decoder. This uses the same P_E, P_T and P_L models as our hybrid generative system, but employs a two-pass decoding scheme to find the max transliteration according to Equation 1. It represents a purely generative solution using otherwise identical architecture.

[6] http://www.microsofttranslator.com

Since our hybrid generative system implements a model that is very similar to those used in phrasal SMT, we also compare against a state-of-the-art phrasal SMT system (Moore and Quirk, 2007). This system is trained by applying the standard SMT pipeline to our Wikipedia title pairs, treating characters as words, using a 7-gram character-level language model, and disabling re-ordering. Unfortunately, the decoder's architecture does not allow the use of a word-level unigram model, reducing the usefulness of this baseline. Instead, we include the target lexicon as a second character-level language model. This baseline indicates the level of performance one can expect by applying phrasal SMT straight out of the box.

Comparing the two baselines qualitatively, both use a combination of generative models inspired by the noisy channel. Sherif and Kondrak employ a word-level unigram model without discriminatively weighting the models, while the Phrasal SMT approach uses weights derived from max-BLEU training without word-level unigrams. The obvious question of what happens when one does both will be answered by our hybrid generative system.

5.2 Development experiments

Table 1 shows development set accuracy for a number of systems and feature types, along with the model size of the corresponding systems, where size is measured in terms of the number of non-zero discriminatively-trained parameters. The accuracy of the Sherif and Kondrak baseline is shown as *SK07*. Despite its lack of discriminative training, word-level unigrams allow the *SK07* baseline to outperform *Phrasal SMT*. In future experiments, we compare only against *SK07*.

The indicator system was tested using only operation indicators, with source context, transition and lexicon indicators added incrementally. All feature types have a substantial impact, with the lexicon providing the boost needed to surpass the baseline. Note that the inclusion of the five frequency bins is sufficient to decrease the overall feature count of the system by 600K, as much fewer mistakes are made during training.

Development of the hybrid generative system used the *SK07* baseline as a starting point. The result of combining its three components into a flat linear model, with all weights set to 1, is shown in Table 1 as *Linear SK07*. This violation of conditional independence assumptions results in a drop in accuracy. However, the *+ perceptron* line shows that setting the three weights with perceptron training results in a huge boost in accuracy, nearly matching our indicator system. Adding features inspired by SMT, such as $P_{E'}(t|s)$, eliminates the gap between the two.

Table 1: Development accuracy and model size

System		Acc.	Size
Baseline	Phrasal SMT	30.7	8
	SK07	33.5	–
Indicator	Operations only	3.6	6.8K
	+ source context	23.9	2.8M
	+ transition	28.6	3.1M
	+ lexicon	**44.2**	2.5M
	+ gen. lexicon	44.1	3.0M
Generative	Linear SK07	31.7	–
	+ perceptron	42.4	3
	+ SMT features	44.1	6
	+ ind. lexicon	**44.3**	12

5.3 Development discussion

Considering their differences, the two systems' proximity in score is quite surprising. Given the character domain's lack of sparsity, and the large amount of available training data, we had expected the hybrid generative system to behave only as a strong baseline; instead, it matched the performance of the indicator system. However, this is not unprecedented: discriminatively weighted generative models have been shown to outperform purely discriminative competitors in various NLP classification tasks (Raina et al., 2004; Toutanova, 2006), and remain the standard approach in statistical translation modeling (Och, 2003).

Examining the development results on an example-by-example basis, we see that the two systems make mostly the same mistakes: for 87% of examples, either both systems are right, or both are wrong. The remainder represents a (relatively small) opportunity to improve through system or feature combination: an oracle that perfectly selects between the two scores 50.6.

One opportunity for straight-forward combination is the target lexicon. Because lexicon frequencies are drawn from an independent word list, and not the transliteration training derivations, there is no reason why both systems cannot use both lexicon representations. Unfortunately, doing so has

Table 2: Test set comparisons

	Wikipedia		MT	
System	Acc.	Top 10	Acc.	Top 10
SK07	33.5	57.9	38.8	57.0
Generative	**43.0**	**65.6**	42.9	**58.3**
Indicator	42.5	63.5	**43.6**	57.7

little impact, as is shown in each system's final row in Table 1. Adding the word unigram model to the indicator system results in slightly lower performance, and a much larger model. Adding the frequency bins to the generative system does improve performance slightly, but attempts to completely replace the generative system's word unigram model with frequency bins resulted in a substantial drop in accuracy.[7]

5.4 Test set comparisons

Table 2 shows the accuracies of the systems selected during development on our testing data. On the held-out Wikipedia examples, the trends observed during development remain the same, with the generative system expanding its lead. Moving to 10-best accuracies changes little, except for slightly narrowing the gap between SK07 and the discriminative systems.

The second column of Table 2 compares the systems on our MT test set. As discussed earlier, this data is quite different from the Wikipedia training set, and as a result, the systems' differences are less pronounced. 1-best accuracy still shows the discriminative systems having a definite advantage, but at the 10-best level, those distinctions are muted.

Compared with the previous work on katakana-to-English transliteration, these accuracies do not look particularly high: both Knight and Graehl (1998) and Bilac and Tanaka (2004) report accuracies above 60% for 1-best transliteration. We should emphasize that this is due to the difficulty of our test data, and that we have tested against a baseline that has been shown to outperform Knight and Graehl (1998). The test data was not filtered for noise, leaving untransliterable cases and loose translations intact. The accuracies reported above are under-estimates of real performance: many transliterations not matching the reference may still be useful to a human reader, such as differences in inflection (e.g., レチノイド [*rechinoido*] — *retinoids*, transliterated as *retinoid*), and spacing (e.g. シエラレオネ [*shierareone*]— *Sierra Leone*, transliterated as *sierraleone*).

[7] Lexicon replacement experiment is not shown in Table 1.

6 Integration with machine translation

We used the transliterations from our indicator system to augment a Japanese-to-English MT system.[8] This treelet-based SMT system (Quirk et al., 2005) is trained on about 4.6M parallel sentence pairs from diverse sources including bilingual books, dictionaries and web publications. Our goal is to measure the impact of machine transliterations on end-to-end translation quality.

6.1 Evaluation method

We use the MT-log translation pairs described in Section 4.2 as a sentence-level translation test set. For each katakana word left untranslated by the baseline SMT engine, we generated 10-best transliteration candidates and added the katakana-English pairs to the SMT system's translation table. Perceptron scores were exponentiated, then normalized, to create probabilities, which were given to the SMT system as $P(source|target)$;[9] all other translation features were set to $\log 1$.

We translated the test set with and without the augmented translation table. 120 sentences were randomly selected from the cases where the translations output by the two SMT systems differed, and were submitted for two types of human evaluation. In the **absolute evaluation**, each SMT output was assigned a score between 1 and 4 (1 = completely useless; 4 = perfect translation); in the **relative evaluation**, the evaluators were presented with a pair of SMT outputs, with and without the transliteration table, and were asked to judge if they preferred one translation over the other. In both evaluation settings, the machine-translated sentences were evaluated by two native speakers of English who have no knowledge of Japanese, with access to a reference translation.

6.2 Results

The evaluation results show that our transliterator does improve the quality of SMT. The BLEU

[8] The human evaluation was carried out before we discovered the effectiveness of the hybrid generative system, but recall that the performance of the two is similar.

[9] The perceptron scores are more naturally interpreted as $P(target|source)$, but the opposite direction is generally the highest-weighted feature in the SMT system's linear model.

Table 3: Relative translation evaluation

	evaluator 1 preference			
eval2pref	+translit	equal	baseline	sum
+translit	**95**	0	2	97
equal	19	**1**	2	22
baseline	1	0	**0**	1
sum	115	1	4	120

score on the entire test set improved only slightly, from 21.8 to 22.0. However, in the absolute human evaluation, the transliteration table increased the average human judgement from 1.5 to 2 out of a maximum score of 4. Table 3 shows the results of the relative evaluation along with the judges' sentence-level agreement. In 95 out of 120 cases, both annotators agreed that the augmented table produced a better translation than the baseline.

One might expect that any replacement of katakana would improve the perception of MT quality. This is not necessarily the case: it can be more confusing to have a drastically incorrect transliteration, such as transliterating アップローダ [*appurooda*] — *uploader* incorrectly as *applaud*. Fortunately, Table 3 shows that we make very few of these sorts of mistakes: the baseline is preferred only rarely. Also note that, according the MT 10-best accuracies in Table 2, we would have expected to improve at most 60% of cases, however, the human judgements indicate that our actual rate of improvement is closer to 80%, which demonstrates that even an imperfect transliteration is often useful.

7 Conclusion

We have presented a discriminative substring decoder for transliteration. Our decoder is based on recent approaches for discriminative character transduction, extended to provide access to a target lexicon. We have presented a comparison of indicator and hybrid generative features in a controlled setting, demonstrating that generative models perform surprisingly well when discriminatively weighted. We have also shown our discriminative models to be superior to a state-of-the-art generative system. Finally, we have demonstrated that machine transliteration is immediately useful to end-to-end SMT.

As mentioned earlier, by focusing on katakana, we bypass the problem of deciding when to transliterate rather than translate; next, we plan to combine our models with a classifier that makes such a decision, allowing us to integrate transliteration into SMT for other language pairs.

References

Yaser Al-Onaizan and Kevin Knight. 2002. Machine transliteration of names in Arabic text. In *ACL Workshop on Comp. Approaches to Semitic Languages*.

Shane Bergsma and Grzegorz Kondrak. 2007. Alignment-based discriminative string similarity. In *ACL*, pages 656–663, Prague, Czech Republic, June.

Slaven Bilac and Hozumi Tanaka. 2004. A hybrid back-transliteration system for Japanese. In *COLING*, pages 597–603, Geneva, Switzerland.

Slaven Bilac and Hozumi Tanaka. 2005. Extracting transliteration pairs from comparable corpora. In *Proceedings of the Annual Meeting of the Natural Language Processing Society*, Japan.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *ACL*, pages 286–293, Morristown, NJ.

Eric Brill, Gary Kacmarcik, and Chris Brockett. 2001. Automatically harvesting katakana-english term pairs from search engine query logs. In *Asia Federation of Natural Language Processing*.

Michael Collins, Brian Roark, and Murat Saraçlar. 2005. Discriminative syntactic language modeling for speech recognition. In *ACL*, pages 507–514, Ann Arbor, USA, June.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *EMNLP*, pages 1080–1089, Honolulu, Hawaii, October.

Dayne Freitag and Shahram Khadivi. 2007. A sequence alignment model based on the averaged perceptron. In *EMNLP*, pages 238–247, Prague, Czech Republic, June.

Ulf Hermjakob, Kevin Knight, and Hal Daumé III. 2008. Name translation in statistical machine translation - learning when to transliterate. In *ACL*, pages 389–397, Columbus, Ohio, June.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *HLT-NAACL*, pages 372–379, Rochester, New York, April.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *ACL*, pages 905–913, Columbus, Ohio, June.

Alexandre Klementiev and Dan Roth. 2006. Named entity transliteration and discovery from multilingual comparable corpora. In *HLT-NAACL*, pages 82–88, New York City, USA, June.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, pages 181–184.

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

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL*.

Haizhou Li, Min Zhang, and Jian Su. 2004. A joint source-channel model for machine transliteration. In *ACL*, pages 159–166, Barcelona, Spain, July.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *COLING-ACL*, pages 761–768, Sydney, Australia, July.

Robert Moore and Chris Quirk. 2007. Faster beam-search decoding for phrasal statistical machine translation. In *MT Summit XI*.

Franz J. Och. 2003. Minimum error rate training for statistical machine translation. In *ACL*, pages 160–167.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *ACL*, pages 271–279, Ann Arbor, USA, June.

Rajat Raina, Yirong Shen, Andrew Y. Ng, and Andrew McCallum. 2004. Classification with hybrid generative/discriminative models. In *Advances in Neural Information Processing Systems 16*.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

Sunita Sarawagi and William Cohen. 2004. Semimarkov conditional random fields for information extraction. In *ICML*.

Tarek Sherif and Grzegorz Kondrak. 2007a. Bootstrapping a stochastic transducer for Arabic-English transliteration extraction. In *ACL*, pages 864–871, Prague, Czech Republic, June.

Tarek Sherif and Grzegorz Kondrak. 2007b. Substring-based transliteration. In *ACL*, pages 944–951, Prague, Czech Republic, June.

Kristina Toutanova. 2006. Competitive generative models with structure learning for nlp classification tasks. In *EMNLP*, pages 576–584, Sydney, Australia, July.

Dmitry Zelenko and Chinatsu Aone. 2006. Discriminative methods for transliteration. In *EMNLP*, pages 612–617, Sydney, Australia, July.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *HLT-NAACL*, pages 257–264, Boston, USA, May.

Hao Zhang, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *ACL*, pages 97–105, Columbus, Ohio, June.

Re-Ranking Models Based-on Small Training Data for Spoken Language Understanding

Marco Dinarelli
University of Trento
Italy
dinarelli@disi.unitn.it

Alessandro Moschitti
University of Trento
Italy
moschitti@disi.unitn.it

Giuseppe Riccardi
University of Trento
Italy
riccardi@disi.unitn.it

Abstract

The design of practical language applications by means of statistical approaches requires annotated data, which is one of the most critical constraint. This is particularly true for Spoken Dialog Systems since considerably domain-specific conceptual annotation is needed to obtain accurate Language Understanding models. Since data annotation is usually costly, methods to reduce the amount of data are needed. In this paper, we show that better feature representations serve the above purpose and that structure kernels provide the needed improved representation. Given the relatively high computational cost of kernel methods, we apply them to just re-rank the list of hypotheses provided by a fast generative model. Experiments with Support Vector Machines and different kernels on two different dialog corpora show that our re-ranking models can achieve better results than state-of-the-art approaches when small data is available.

1 Introduction

Spoken Dialog Systems carry out automatic speech recognition and shallow natural language understanding by heavily relying on statistical models. These in turn need annotated data describing the application domain. Such annotation is far the most expensive part of the system design. Therefore, methods reducing the amount of labeled data can speed up and lower the overall amount of work.

Among others, Spoken Language Understanding (SLU) is an important component of the systems above, which requires training data to translate a spoken sentence into its meaning representation based on semantic constituents. These are conceptual units instantiated by sequences of words.

In the last decade two major approaches have been proposed to automatically map words in concepts: (i) generative models, whose parameters refer to the joint probability of concepts and constituents; and (ii) discriminative models, which learn a classification function based on conditional probabilities of concepts given words.

A simple but effective generative model is the one based on Finite State Transducers. It performs SLU as a translation process from words to concepts using Finite State Transducers (FST). An example of discriminative model used for SLU is the one based on Support Vector Machines (SVMs) (Vapnik, 1995), as shown in (Raymond and Riccardi, 2007). In this approach, data is mapped into a vector space and SLU is performed as a classification problem using Maximal Margin Classifiers (Vapnik, 1995). A relatively more recent approach for SLU is based on Conditional Random Fields (CRF) (Lafferty et al., 2001). CRFs are undirected graphical and discriminative models. They use conditional probabilities to account for many feature dependencies without the need of explicitly representing such dependencies.

Generative models have the advantage to be more robust to overfitting on training data, while discriminative models are more robust to irrelevant features. Both approaches, used separately, have shown good accuracy (Raymond and Riccardi, 2007), but they have very different characteristics and the way they encode prior knowledge is very different, thus designing models that take into account characteristics of both approaches are particularly promising.

In this paper, we propose a method for SLU based on generative and discriminative models: the former uses FSTs to generate a list of SLU hypotheses, which are re-ranked by SVMs. To effectively design our re-ranker, we use all pos-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1076–1085,
Singapore, 6-7 August 2009.

sible word/concept subsequences with gaps of the spoken sentence as features (*i.e.* all possible n-grams). Gaps allow for encoding long distance dependencies between words in relatively small sequences. Since the space of such features is huge, we adopted kernel methods, i.e. sequence kernels (Shawe-Taylor and Cristianini, 2004) and tree kernels (Collins and Duffy, 2002; Moschitti, 2006a) to implicitly encode them along with other structural information in SVMs.

We experimented with different approaches for training the discriminative models and two different corpora: the french MEDIA corpus (Bonneau-Maynard et al., 2005) and a corpus made available by the European project LUNA[1] (Dinarelli et al., 2009b). In particular, the new contents with respect to our previous work (Dinarelli et al., 2009a) are:

- We designed a new sequential structure (SK2) and two new hierarchical tree structures (MULTILEVEL and FEATURES) for re-ranking models (see Section 4.2). The latter combined with two different tree kernels originate four new different models.
- We experimented with automatic speech transcriptions thus assessing the robustness to noise of our models.
- We compare our models against Conditional Random Field (CRF) approaches described in (Hahn et al., 2008), which are the current state-of-the-art in SLU. Learning curves clearly show that our models improve CRF, especially when small data sets are used.

The remainder of the paper is organized as follows: Section 2 introduces kernel methods for structured data, Section 3 describes the generative model producing the initial hypotheses whereas Section 4 presents the discriminative models for re-ranking them. The experiments and results are reported in Section 5 and the conclusions are drawn in Section 6.

2 Feature Engineering via Structure Kernels

Kernel methods are viable approaches to engineer features for text processing, *e.g.* (Collins and Duffy, 2002; Kudo and Matsumoto, 2003; Cumby and Roth, 2003; Cancedda et al., 2003; Culotta and Sorensen, 2004; Toutanova et al., 2004; Kudo et al., 2005; Moschitti, 2006a; Moschitti et al., 2007; Moschitti, 2008; Moschitti et al., 2008; Moschitti and Quarteroni, 2008). In the following, we describe structure kernels, which will be used to engineer features for our discriminative reranker.

2.1 String Kernels

The String Kernels that we consider count the number of substrings containing gaps shared by two sequences, *i.e.* some of the symbols of the original string are skipped. We adopted the efficient algorithm described in (Shawe-Taylor and Cristianini, 2004; Lodhi et al., 2000). More specifically, we used words and markers as symbols in a style similar to (Cancedda et al., 2003; Moschitti, 2008). For example, given the sentence: *How may I help you ?* sample substrings, extracted by the Sequence Kernel (SK), are: *How help you ?*, *How help ?*, *help you*, *may help you*, etc.

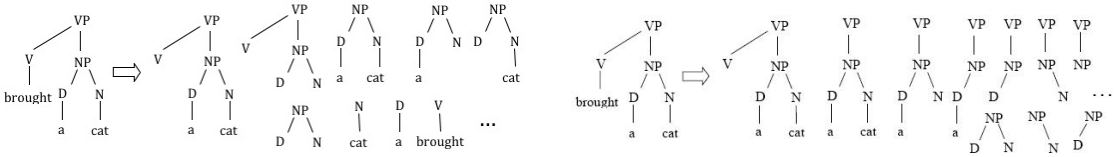
2.2 Tree kernels

Tree kernels represent trees in terms of their substructures (fragments). The kernel function detects if a tree subpart (common to both trees) belongs to the feature space that we intend to generate. For such purpose, the desired fragments need to be described. We consider two important characterizations: the syntactic tree (STF) and the partial tree (PTF) fragments.

2.2.1 Tree Fragment Types

An STF is a general subtree whose leaves can be non-terminal symbols (also called SubSet Tree (SST) in (Moschitti, 2006a)). For example, Figure 1(a) shows 10 STFs (out of 17) of the subtree rooted in `VP` (of the left tree). The STFs satisfy the constraint that grammatical rules cannot be broken. For example, `[VP [V NP]]` is an STF, which has two non-terminal symbols, `V` and `NP`, as leaves whereas `[VP [V]]` is not an STF. If we relax the constraint over the STFs, we obtain more general substructures called *partial trees fragments* (PTFs). These can be generated by the application of partial production rules of the grammar, consequently `[VP [V]]` and `[VP [NP]]` are valid PTFs. Figure 1(b) shows that the number of PTFs derived from the same tree as before is still higher (*i.e.* 30 PTs).

[1] Contract n. 33549



(a) Syntactic Tree fragments (STF)

(b) Partial Tree fragments (PTF)

Figure 1: Examples of different classes of tree fragments.

2.3 Counting Shared Subtrees

The main idea of tree kernels is to compute the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. To evaluate the above kernels between two T_1 and T_2, we need to define a set $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$, *i.e.* a tree fragment space and an indicator function $I_i(n)$, equal to 1 if the target f_i is rooted at node n and equal to 0 otherwise. A tree-kernel function over T_1 and T_2 is $TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1) I_i(n_2)$. The latter is equal to the number of common fragments rooted in the n_1 and n_2 nodes.

The algorithm for the efficient evaluation of Δ for the syntactic tree kernel (STK) has been widely discussed in (Collins and Duffy, 2002) whereas its fast evaluation is proposed in (Moschitti, 2006b), so we only describe the equations of the partial tree kernel (PTK).

2.4 The Partial Tree Kernel (PTK)

PTFs have been defined in (Moschitti, 2006a). Their computation is carried out by the following Δ function:

1. if the node labels of n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
2. else $\Delta(n_1, n_2) =$
$1 + \sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1) = l(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))$

where $\vec{I}_1 = \langle h_1, h_2, h_3, .. \rangle$ and $\vec{I}_2 = \langle k_1, k_2, k_3, .. \rangle$ are index sequences associated with the ordered child sequences c_{n_1} of n_1 and c_{n_2} of n_2, respectively, $\vec{I}_{1j}$ and $\vec{I}_{2j}$ point to the j-th child in the corresponding sequence, and, again, $l(\cdot)$ returns the sequence length, *i.e.* the number of children.

Furthermore, we add two decay factors: μ for the depth of the tree and λ for the length of the child subsequences with respect to the original sequence, *i.e.* we account for gaps. It follows that $\Delta(n_1, n_2) =$

$$\mu\Big(\lambda^2 + \sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1) = l(\vec{I}_2)} \lambda^{d(\vec{I}_1) + d(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\Big), \quad (1)$$

where $d(\vec{I}_1) = \vec{I}_{1l(\vec{I}_1)} - \vec{I}_{11}$ and $d(\vec{I}_2) = \vec{I}_{2l(\vec{I}_2)} - \vec{I}_{21}$. This way, we penalize both larger trees and child subsequences with gaps. Eq. 1 is more general than the Δ equation for STK. Indeed, if we only consider the contribution of the longest child sequence from node pairs that have the same children, we implement STK.

3 Generative Model: Stochastic Conceptual Language Model (SCLM)

The first step of our approach is to produce a list of SLU hypotheses using a Stochastic Conceptual Language Model. This is the same described in (Raymond and Riccardi, 2007) with the only difference that we train the language model using the SRILM toolkit (Stolcke, 2002) and we then convert it into a Stochastic Finite State Transducer (SFST). Such method allows us to use a wide group of language models, backed-off or interpolated with many kind of smoothing techniques (Chen and Goodman, 1998).

To exemplify our SCLM let us consider the following input italian sentence taken from the LUNA corpus along with its English translation:

Ho un problema col monitor.
(I have a problem with my screen).

A possible semantic annotation is:

null{*ho*} ***PROBLEM***{*un problema*} ***HARDWARE***{*col monitor*},

where ***PROBLEM*** and ***HARDWARE*** are two domain concepts and ***null*** is the label used for words not meaningful for the task. To associate word sequences with concepts, we use *begin*

(*B*) and *inside* (*I*) markers after each word of a sequence, *e.g.*:

null$\{ho\}$ ***PROBLEM-B***$\{un\}$ ***PROBLEM-I***$\{problema\}$ ***HARDWARE-B***$\{col\}$ ***HARDWARE-I***$\{monitor\}$

This annotation is automatically performed by a model based on a combination of three transducers:

$$\lambda_{SLU} = \lambda_W \circ \lambda_{W2C} \circ \lambda_{SLM},$$

where λ_W is the transducer representation of the input sentence, λ_{W2C} is the transducer mapping words to concepts and λ_{SLM} is the Stochastic Conceptual Language Model trained with SRILM toolkit and converted in FST. The SCLM represents joint probability of word and concept sequences by using the joint probability:

$$P(W, C) = \prod_{i=1}^{k} P(w_i, c_i | h_i),$$

where $W = w_1..w_k$, $C = c_1..c_k$ and $h_i = w_{i-1}c_{i-1}..w_1c_1$.

4 Discriminative re-ranking

Our discriminative re-ranking is based on SVMs trained with pairs of conceptually annotated sentences produced by the FST-based generative model described in the previous section. An SVM learn to classify which annotation has an error rate lower than the others so that it can be used to sort the *m*-best annotations based on their correctness. While for SVMs details we remaind to the wide literature available, for example (Vapnik, 1995) or (Shawe-Taylor and Cristianini, 2004), in this section we focus on hypotheses generation and on the kernels used to implement our re-ranking model.

4.1 Generation of m-best concept labeling

Using the FST-based model described in Section 3, we can generate the list of *m* best hypotheses ranked by the joint probability of the Stochastic Conceptual Language Model (SCLM). The Re-ranking model proposed in this paper re-ranks such list.

After an analysis of the *m*-best hypothesis list, we noticed that many times the first hypothesis ranked by SCLM is not the most accurate, *i.e.* the error rate evaluated with its Levenshtein distance from the manual annotation is not the lowest among the *m* hypotheses. This means that re-ranking hypotheses could improve the SLU accuracy. Intuitively, to achieve satisfactory results, different features from those used by SCLM should be considered to exploit in a different way the information encoded in the training data.

4.2 Structural features for re-ranking

The kernels described in previous sections provide a powerful technology for exploiting features of structured data. These kernels were originally designed for data annotated with syntactic parse trees. In Spoken Language Understanding the data available are text sentences with their semantic annotation based on basic semantic constituents. This kind of data has a rather flat structure with respect to syntactic parse trees. Thus, to exploit the power of kernels, a careful design of the structures used to represent data must be carried out, where the goal is to build tree-like annotation from the semantic annotation. For this purpose, we note that the latter is made upon sentence chunks, which implicitly define syntactic structures as long as the annotation is consistent in the corpus.

We took into account the characteristics of the presented kernels and the structure of semantic annotated data. As a result we designed the tree structures shown in figures 2(a), 2(b) and 3 for STK and PTK and sequential structures for SK defined in the following (where all the structures refer to the same example presented in Section 3, *i.e. Ho un problema col monitor*). The structures used with SK are:

($SK1$) ***NULL*** *ho* ***PROBLEM-B*** *un* ***PROBLEM-I*** *problema* ***HARDWARE-B*** *col* ***HARDWARE-I*** *monitor*

($SK2$) ***NULL*** *ho* ***PROBLEM B*** *un* ***PROBLEM I*** *problema* ***HARDWARE B*** *col* ***HARDWARE I*** *monitor*,

For simplicity, from now on, the two structures will be referred as *SK1* and *SK2* (String Kernel 1 and 2). They differer in the use of chunk markers *B* and *I*. In *SK1*, markers are part of the concept, thus they increase the number of semantic tags in the data whereas in *SK2* markers are put apart as separated words so that they can mark effectively the beginning and the end of a concept, but for the same reason they can add noise in the sentence. Notice that the order of words and concepts is meaningful since each word is preceded by its corresponding concepts.

The structures shown in Figure 2(a), 2(b) and 3

have been designed for STK and PTK. They provide trees with increasing structure complexity as described in the following.

The first structure (FLAT) is a simple tree providing direct dependency between words and chunked concepts. From it, STK and PTK can extract relevant features (tree fragments).

The second structure (MULTILEVEL) has one more level of nodes and yields the same separation of concepts and markers shown in *SK1*. Notice that the same separation can be carried out putting the markers *B* and *I* as features at the same level of the words. This would increase exponentially (in the number of leaves) the number of subtrees taken into account by the STK computation. Since STK doesn't separate children, as described in Section 2.3, the structure we chose is lighter but also more rigid.

The third structure (FEATURES) is a more complex structure. It allows to use a wide number of features (like Word categories, POS tags, morpho-syntactic features), which are commonly used in this kind of task. As described above, the use of features exponentially increases the number of subtrees taken into account by kernel computations but they also increase the robustness of the model. In this work we only used Word Categories as features. They are domain independent, e.g. *"Months"*, *"Dates"*, *"Number"* etc. or POS tags, which are useful to generalize target words. Note also that the features in common between two trees must appear in the same child-position, hence we sort them based on their indices, *e.g.'F0'* for words and *'F1'* for word categories.

4.3 Re-ranking models using sequences

The FST generates the *m* most likely concept annotations. These are used to build annotation pairs, $\langle s^i, s^j \rangle$, which are positive instances if s^i has a lower concept annotation error than s^j, with respect to the manual annotation. Thus, a trained binary classifier can decide if s^i is more accurate than s^j. Each candidate annotation s^i is described by a word sequence with its concept annotation. Considering the example in the previous section, a pair of annotations $\langle s^i, s^j \rangle$ could be

s^i: ***NULL*** *ho* ***PROBLEM-B*** *un* ***PROBLEM-I*** *problema* ***HARDWARE-B*** *col* ***HARDWARE-I*** *monitor*

s^j: ***NULL*** *ho* ***ACTION-B*** *un* ***ACTION-I*** *problema* ***HARDWARE-B*** *col* ***HARDWARE-B*** *monitor*

where **NULL**, **ACTION** and **HARDWARE** are the assigned concepts. The second annotation is less accurate than the first since *problema* is erroneously annotated as ***ACTION*** and *"col monitor"* is split in two different concepts.

Given the above data, the sequence kernel is used to evaluate the number of common *n*-grams between s^i and s^j. Since the string kernel skips some elements of the target sequences, the counted *n*-grams include: concept sequences, word sequences and any subsequence of words and concepts at any distance in the sentence.

Such counts are used in our re-ranking function as follows: let e_k be the pair $\langle s_k^1, s_k^2 \rangle$ we evaluate the kernel:

$$\begin{aligned} K_R(e_1, e_2) &= SK(s_1^1, s_2^1) + SK(s_1^2, s_2^2) \\ &- SK(s_1^1, s_2^2) - SK(s_1^2, s_2^1) \end{aligned} \quad (2)$$

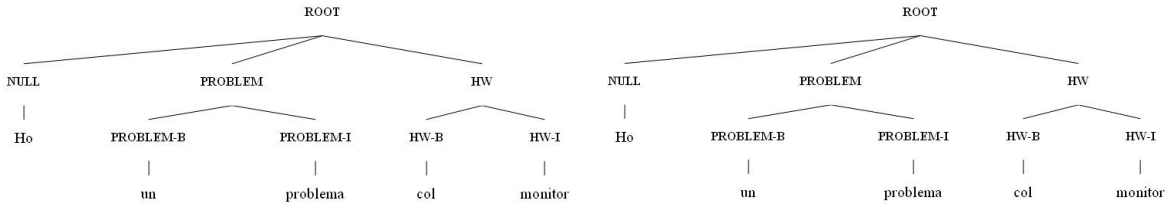
This schema, consisting in summing four different kernels, has been already applied in (Collins and Duffy, 2002; Shen et al., 2003) for syntactic parsing re-ranking, where the basic kernel was a tree kernel instead of SK. It was also used also in (Shen et al., 2004) to re-rank different candidates of the same hypothesis for machine translation. Notice that our goal is different from the one tackled in such paper and, in general, it is more difficult: we try to learn which is the best annotation of a given input sentence, while in (Shen et al., 2004), they learn to distinguish between *"good"* and *"bad"* translations of a sentence. Even if our goal is more difficult, our approach is very effective, as shown in (Dinarelli et al., 2009a). It is more appropriate since in parse re-ranking there is only one best hypothesis, while in machine translation a sentence can have more than one correct translations.

Additionally, in (Moschitti et al., 2006; Moschitti et al., 2008) a tree kernel was applied to semantic trees similar to the one introduced in the next section to re-rank Semantic Role Labeling annotations.

4.4 Re-ranking models using trees

Since the aim of concept annotation re-ranking is to exploit innovative and effective source of information, we can use, in addition to sequence kernels, the power of tree kernels to generate correlation between concepts and word structures.

Figures 2(a), 2(b) and 3 describe the structural association between the concept and the word



(a) *FLAT* Tree (b) *MULTILEVEL* Tree

Figure 2: Examples of structures used for STK and PTK

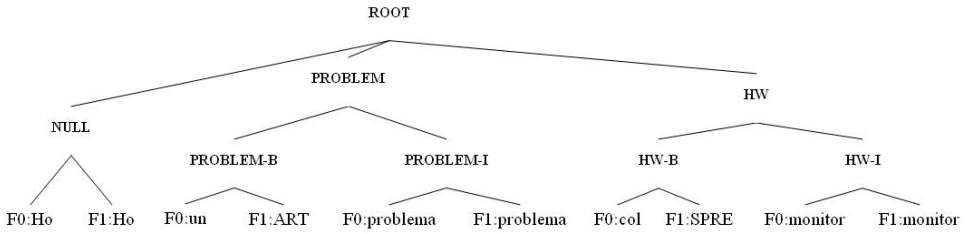


Figure 3: The *FEATURES* semantic tree used for STK or PTK

Corpus LUNA	Train set		Test set	
	words	concepts	words	concepts
Dialogs	183		67	
Turns	1.019		373	
Tokens	8.512	2.887	2.888	984
Vocab.	1.172	34	-	-
OOV rate	-	-	3.2%	0.1%

Table 1: Statistics on the LUNA corpus

Corpus Media	Train set		Test set	
	words	concepts	words	concepts
Turns	12,922		3,518	
# of tokens	94,912	43,078	26,676	12,022
Vocabulary	5,307	80	-	-
OOV rate	-	-	0.01%	0.0%

Table 2: Statistics on the MEDIA corpus

level. This kind of trees allows us to engineer new kernels and consequently new features (Moschitti et al., 2008), *e.g.* their subparts extracted by STK or PTK, like the tree fragments in figures 1(a) and 1(b). These can be used in SVMs to learn the classification of words in concepts.

More specifically, in our approach, we use tree fragments to establish the order of correctness between two alternative annotations. Therefore, given two trees associated with two annotations, a re-ranker based on tree kernel can be built in the same way of the sequence-based kernel by substituting SK in Eq. 2 with STK or PTK. The major advantage of using trees is the hierarchical dependencies between its nodes, allowing for the use of richer n-grams with back-off models.

5 Experiments

In this section, we describe the corpora, parameters, models and results of our experiments on re-ranking for SLU. Our baseline is constituted by the error rate of systems solely based on either FST or SVMs. The re-ranking models are built on the FST output, which in turn is applied to both manual or automatic transcriptions.

5.1 Corpora

We used two different speech corpora:

The LUNA corpus, produced in the homonymous European project, is the first Italian dataset of spontaneous speech on spoken dialogs. It is based on help-desk conversations in a domain of software/hardware repairing (Dinarelli et al., 2009b). The data is organized in transcriptions and annotations of speech based on a new multi-level protocol. Although data acquisition is still in progress, 250 dialogs have been already acquired with a WOZ approach and other 180 Human-Human (HH) dialogs have been annotated. In this work, we only use WOZ dialogs, whose statistics are reported in Table 1.

The corpus MEDIA was collected within the French project MEDIA-EVALDA (Bonneau-Maynard et al., 2005) for development and evaluation of spoken understanding models and linguistic studies. The corpus is composed of 1257 dialogs (from 250 different speakers) acquired with a Wizard of Oz (WOZ) approach in the context of hotel room reservations and tourist information.

Statistics on transcribed and conceptually annotated data are reported in Table 2.

5.2 Experimental setup

Given the small size of LUNA corpus, we did not carried out any parameterization thus we used default or a priori parameters. We experimented with LUNA and three different re-rankers obtained with the combination of SVMs with STK, PTK and SK, described in Section 4. The initial annotation to be re-ranked is the list of the ten best hypotheses output by an FST model.

We point out that, on the large Media dataset the processing time is considerably high[2] so we could not run all the models.

We trained all the SCLMs used in our experiments with the SRILM toolkit (Stolcke, 2002) and we used an interpolated model for probability estimation with the Kneser-Ney discount (Chen and Goodman, 1998). We then converted the model in an FST again with SRILM toolkit.

The model used to obtain the SVM baseline for concept classification was trained using YamCHA (Kudo and Matsumoto, 2001). As re-ranking models based on structure kernels and SVMs, we used the SVM-Light-TK toolkit (available at disi.unitn.it/moschitti). For λ (see Section 3), cost-factor and trade-off parameters, we used, 0.4, 1 and 1, respectively (*i.e.* the default parameters). The number m of hypotheses was always set to 10.

The CRF model we compare with was trained with the CRF++ tool, available at http://crfpp.sourceforge.net/. The model is equivalent to the one described in (Hahn et al., 2008). As features, we used word and morpho-syntactic categories in a window of [-2, +2] with respect to the current token, plus bigrams of concept tags (see (Hahn et al., 2008) and the CRF++ web site for more details).

Such model is very effective for SLU. In (Hahn et al., 2008), it is compared with other four models (Stochastic Finite State Transducers, Support Vector Machines, Machine Translation, Positional-Based Log-linear model) and it is by far the best on MEDIA. Additionally, in (Raymond and Riccardi, 2007), a similar CRF model was compared with FST and SVMs on ATIS and on a different version of MEDIA, showing again to be very effective.

[2]The number of parameters of the models and the number of training approaches make the exhaustive experimentation very expensive in terms of processing time, which would be roughly between 2 and 3 months of a typical workstation.

Structure	STK	PTK	SK
FLAT	18.5	19.3	-
MULTILEVEL	20.6	19.1	-
FEATURES	19.9	18.4	-
SK1	-	-	16.2
SK2	-	-	18.5

Table 3: CER of SVMs using STK, PTK and SK on LUNA (manual transcriptions). The Baselines, FST and SVMs alone, show a CER of **23.2%** and **26.3%**, respectively.

Model	MEDIA (CER)	LUNA (CER)
FST	13.7%	23.2%
CRF	11.5%	20.4%
SVM-RR (PTK)	12.1%	18.4%

Table 4: Results of SLU experiments on MEDIA and LUNA test set (manual transcriptions).

We ran SLU experiments on manual and automatic transcriptions. The latter are produced by a speech recognizer with a WER of 41.0% and 31.4% on the LUNA and the MEDIA test sets, respectively.

5.3 Training approaches

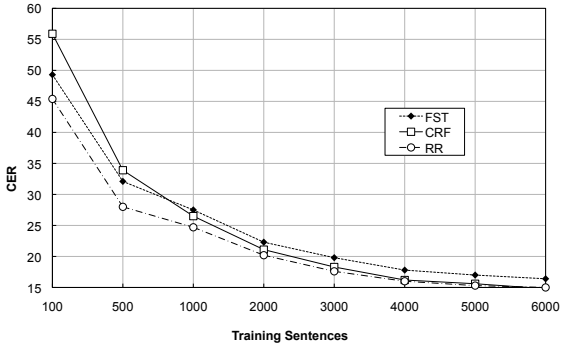
The FST model generates the *10*-best annotations, *i.e.* the data used to train the re-ranker based on SVMs. Different training approaches can be carried out based on the use of the data. We divided the training set in two parts. We train FSTs on part 1 and generate the *10*-best hypotheses using part 2, thus providing the first chunk of re-ranking data. Then, we re-apply these steps inverting part 1 with part 2 to provide the second data chunk. Finally, we train the re-ranker on the merged data.

For classification, we generate the *10*-best hypotheses of the whole test set using the FST trained on all training data.

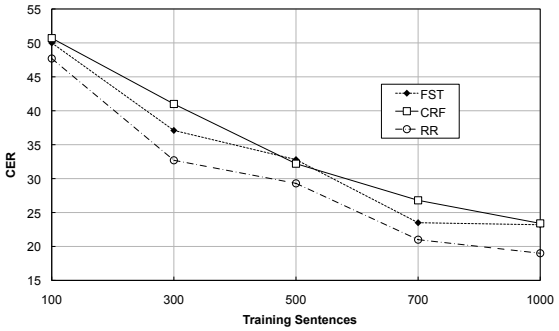
5.4 Re-ranking results

In Tables 3, 4 and 5 and Figures 4(a) and 4(b) we report the results of our experiments, expressed in terms of concept error rate (CER). CER is a standard measure based on the Levensthein alignment of sentences and it is computed as the ratio between inserted, deleted and confused concepts and the number of concepts in the reference sentence.

Table 3 shows the results on the LUNA corpus using the different training approaches, kernels and structures described in this paper. The



(a) Learning Curve on MEDIA corpus using the RR model based on SVMs and STK



(b) Learning Curve on LUNA corpus using the RR model based on SVMs and SK

Figure 4: Learning curves on MEDIA and LUNA corpora using FST, CRF and RR on the FST hypotheses

Model	MEDIA (CER)	LUNA (CER)
FST	28.6%	42.7%
CRF	24.0%	41.8%
SVM-RR (PTK)	25.0%	38.9%

Table 5: Results of SLU experiments on MEDIA and LUNA test set (automatic transcriptions with a WER 31.4% on MEDIA and 41% on LUNA)

dash symbol means that the structure cannot be applied to the corresponding kernel. We note that our re-rankers significantly improve our baselines, *i.e.* 23.2% CER for FST and 26.3% CER for SVM concept classifiers. For example, SVM re-ranker using SK, in the best case, improves FST concept classifier of 23.2-16.2 = 7 points.

Note also that the structures designed for trees yield quite different results depending on which kernel is used. We can see in Table 3 that the best result using STK is obtained with the simplest structure (FLAT), while with PTK the best result is achieved with the most complex structure (FEATURES). This is due to the fact that STK does not split the children of each node, as explained in Section 2.2, and so structures like *MULTILEVEL* and *FEATURES* are too rigid and prevent the STK to be effective.

For lack of space we do not report all the results using different kernels and structures on MEDIA, but we underline that as MEDIA is a more complex task (34 concepts in LUNA, 80 in MEDIA), the more complex structures are more effective to capture word-concept dependencies and the best results were obtained using the *FEATURES* tree.

Table 4 shows the results of the SLU experiments on the MEDIA and LUNA test sets using the manual transcriptions of spoken sentences and a re-ranker based on PTK and the *FEATURES* structure (already reported in the previous table). We used PTK since it is enough efficient to carry out the computation on the much larger Media corpus although as previously shown it is less accurate than SK.

We note that on a big corpus like MEDIA, the baseline models (FST and CRF) can be accurately learned thus less errors can be "corrected". As a consequence, our re-ranking approach does not improve CRF but it still improves the FSTs baseline of 1.6% points (11.7% of relative improvement).

The same behavior is reproduced for the SLU experiments on automatic transcriptions, shown in Table 5. We note that, on the LUNA corpus, CRFs are more accurate than FSTs (0.9% points), but they are significantly improved by the re-ranking model (2.9% points), which also improves the FSTs baseline by 3.8% points. On the MEDIA corpus, the re-ranking model is again very accurate improving the FSTs baseline of 3.6% points (12.6% relative improvement) on attribute annotation, but the most accurate model is again CRF (1% points better than the re-ranking model).

5.5 Discussion

The different behavior of the re-ranking model in the LUNA and MEDIA corpora is due partially to the task complexity, but it is mainly due to the fact that CRFs have been deeply studied and experimented (see (Hahn et al., 2008)) on MEDIA. Thus CRF parameters and features have been largely optimized. We believe that the re-ranking model can be relevantly improved by carrying out parameter optimization and new structural feature de-

sign.

Moreover, our re-ranking models achieve the highest accuracy for automatic concept annotation when small data sets are available. To show this, we report in Figure 4(a) and 4(b) the learning curves according to an increasing number of training sentences on the MEDIA and LUNA corpora, respectively. To draw the first plot, we used a re-ranker based on STK (and the FLAT tree), which is less accurate than the other kernels but also the most efficient in terms of training time. In the second plot, we report the re-ranker accuracy using SK applied to *SK1* structure.

In these figures, the FST baseline performance is compared with our re-ranking (RR) and a Conditional Random Field (CRF) model. The above curves clearly shows that for small datasets our RR model is better than CRF whereas when the data increases, CRF accuracy approaches the one of the RR.

Regarding the use of kernels two main findings can be derived:

- Kernels producing a high number of features, *e.g.* SK, produce accuracy higher than kernels less rich in terms of features, *i.e.* STK. In particular STK is improved by 18.5-16.2=2.3 points (Table 3). This is an interesting result since it shows that (a) a kernel producing more features also produces better re-ranking models and (b) kernel methods give a remarkable help in feature design.

- Although the training data is small, the re-rankers based on kernels appear to be very effective. This may also alleviate the burden of annotating large amount of data.

6 Conclusions

In this paper, we propose discriminative re-ranking of concept annotation to jointly exploit generative and discriminative models. We improve the FST-based generative approach, which is a state-of-the-art model in LUNA, by 7 points, where the more limited availability of annotated data leaves a larger room for improvement. Our re-ranking model also improves FST and CRF on MEDIA when small data sets are used.

Kernel methods show that combinations of feature vectors, sequence kernels and other structural kernels, *e.g.* on shallow or deep syntactic parse trees, appear to be a promising future research line[3]. Finally, the experimentation with automatic speech transcriptions revealed that to test the robustness of our models to transcription errors.

In the future we would like to extend this research by focusing on advanced shallow semantic approaches such as predicate argument structures, *e.g.* (Giuglea and Moschitti, 2004; Moschitti and Cosmin, 2004; Moschitti et al., 2008). Additionally, term similarity kernels, *e.g.* (Basili et al., 2005; Bloehdorn et al., 2006), will be likely improve our models, especially when combined syntactic and semantic kernels are used, i.e. (Bloehdorn and Moschitti, 2007a; Bloehdorn and Moschitti, 2007b).

References

Roberto Basili, Alessandro Moschitti, and Maria Teresa Pazienza. 1999. A text classifier based on linguistic processing. In *Proceedings of IJCAI 99, Machine Learning for Information Filtering*.

Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2005. Effective use of WordNet semantics via kernel-based learning. In *Proceedings of CoNLL-2005*, Ann Arbor, Michigan.

Stephan Bloehdorn and Alessandro Moschitti. 2007a. Combined syntactic and semantic kernels for text classification. In *Proceedings of ECIR 2007, Rome, Italy*.

Stephan Bloehdorn and Alessandro Moschitti. 2007b. Structure and semantics for expressive text kernels. In *In proceedings of CIKM '07*.

Stephan Bloehdorn, Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2006. Semantic kernels for text classification based on topological measures of feature similarity. In *Proceedings of ICDM 06, Hong Kong, 2006*.

H. Bonneau-Maynard, S. Rosset, C. Ayache, A. Kuhn, and D. Mostefa. 2005. Semantic annotation of the french media dialog corpus. In *Proceedings of Interspeech2005*, Lisbon, Portugal.

N. Cancedda, E. Gaussier, C. Goutte, and J. M. Renders. 2003. Word sequence kernels. *J. Mach. Learn. Res.*, 3.

S. F. Chen and J. Goodman. 1998. An empirical study of smoothing techniques for language modeling. In *Technical Report of Computer Science Group*, Harvard, USA.

[3] A basic approach is the use of part-of-speech tags like for example in text categorization (Basili et al., 1999) but given the high efficiency of modern syntactic parsers we can use the complete parse tree.

M. Collins and N. Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete structures, and the voted perceptron. In *ACL02*, pages 263–270.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of ACL'04*.

Chad Cumby and Dan Roth. 2003. Kernel Methods for Relational Learning. In *Proceedings of ICML 2003*.

Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. 2009a. Re-ranking models for spoken language understanding. In *Proceedings of EACL2009*, Athens, Greece.

Marco Dinarelli, Silvia Quarteroni, Sara Tonelli, Alessandro Moschitti, and Giuseppe Riccardi. 2009b. Annotating spoken dialogs: from speech segments to dialog acts and frame semantics. In *Proceedings of SRSL 2009 Workshop of EACL*, Athens, Greece.

Ana-Maria Giuglea and Alessandro Moschitti. 2004. Knowledge Discovery using Framenet, Verbnet and Propbank. In A. Meyers, editor, *Workshop on Ontology and Knowledge Discovering at ECML 2004*, Pisa, Italy.

Stefan Hahn, Patrick Lehnen, Christian Raymond, and Hermann Ney. 2008. A comparison of various methods for concept tagging for spoken language understanding. In *Proceedings of LREC*, Marrakech, Morocco.

T. Kudo and Y. Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of NAACL2001*, Pittsburg, USA.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of ACL'03*.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proceedings of ACL'05*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML2001*, US.

Huma Lodhi, John S. Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. 2000. Text classification using string kernels. In *NIPS*.

Alessandro Moschitti and Adrian Bejan Cosmin. 2004. A semantic kernel for predicate argument classification. In *CoNLL-2004*, Boston, MA, USA.

Alessandro Moschitti and Silvia Quarteroni. 2008. Kernels on linguistic structures for answer extraction. In *Proceedings of ACL-08: HLT, Short Papers*, Columbus, Ohio.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2006. Semantic role labeling via tree kernel joint inference. In *Proceedings of CoNLL-X*, New York City.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proceedings of ACL'07*, Prague, Czech Republic.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Alessandro Moschitti. 2006a. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In *Proceedings of ECML 2006*, pages 318–329, Berlin, Germany.

Alessandro Moschitti. 2006b. Making Tree Kernels Practical for Natural Language Learning. In *Proceedings of EACL2006*.

Alessandro Moschitti. 2008. Kernel methods, syntax and semantics for relational text categorization. In *Proceeding of CIKM '08*, NY, USA.

C. Raymond and G. Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Proceedings of Interspeech2007*, Antwerp,Belgium.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Libin Shen, Anoop Sarkar, and Aravind k. Joshi. 2003. Using LTAG Based Features in Parse Reranking. In *Proceedings of EMNLP'06*.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *HLT-NAACL*, pages 177–184.

A. Stolcke. 2002. Srilm: an extensible language modeling toolkit. In *Proceedings of SLP2002*, Denver, USA.

Kristina Toutanova, Penka Markova, and Christopher Manning. 2004. The Leaf Path Projection View of Parse Trees: Exploring String Kernels for HPSG Parse Selection. In *Proceedings of EMNLP 2004*.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

Empirical Exploitation of Click Data for Task Specific Ranking

Anlei Dong Yi Chang Shihao Ji Ciya Liao Xin Li Zhaohui Zheng
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94089
{anlei,yichang,shihao,ciyaliao,xinli,zhaohui}@yahoo-inc.com

Abstract

There have been increasing needs for task specific rankings in web search such as rankings for specific query segments like long queries, time-sensitive queries, navigational queries, etc; or rankings for specific domains/contents like answers, blogs, news, etc. In the spirit of "divide-and-conquer", task specific ranking may have potential advantages over generic ranking since different tasks have task-specific features, data distributions, as well as feature-grade correlations. A critical problem for the task-specific ranking is training data insufficiency, which may be solved by using the data extracted from click log. This paper empirically studies how to appropriately exploit click data to improve rank function learning in task-specific ranking. The main contributions are 1) the exploration on the utilities of two promising approaches for click pair extraction; 2) the analysis of the role played by the noise information which inevitably appears in click data extraction; 3) the appropriate strategy for combining training data and click data; 4) the comparison of click data which are consistent and inconsistent with baseline function.

1 Introduction

Learning-to-rank approaches (Liu, 2008) have been widely applied in commercial search engines, in which ranking models are learned using labeled documents. Significant efforts have been made in attempt to learn a generic ranking model which can appropriately rank documents for all queries . However, web users' query intentions are extremely heterogeneous, which makes it difficult for a generic ranking model to achieve best ranking results for all queries. For this reason, there have been increasing needs for task specific rankings in web search such as rankings for specific query segments like long queries, time-sensitive queries, navigational queries, etc; or rankings for specific domains/contents like answers, blogs, news, etc. Therefore, a specific ranking task usually correspond to a category of queries; when the search engine determines that a query is belonging to this category, it will call the ranking function dedicated to this ranking task. The motivation of this divide-and-conquer strategy is that, task specific ranking may have potential advantages over generic ranking since different tasks have task-specific features, data distributions, as well as feature-grade correlations.

Such a dedicated ranking model can be trained using the labeled data belonging to this query category (which is called dedicated training data). However, the amount of training data dedicated to a specific ranking task is usually insufficient because human labeling is expensive and time-consuming, not to mention there are multiple ranking tasks that need to be taken care of. To deal with the training data insufficiency problem for task-specific ranking, we propose to extract click-through data and incorporate it with dedicated training data to learn a dedicated model.

In order to incorporate click data to improve the ranking for a dedicate query category, it is critical to fully exploit click information. We empirically explore the related approaches for the appropriate click data exploitation in task-specific rank function learning. Figure 1 illustrates the procedures and critical components to be studied.

1) Click data mining: the purpose is to extract informative and reliable users' preference information from click log. We employ two promising approaches: one is heuristic rule approach, the other is sequential supervised learning approach.

2) Sample selection and combination: with labeled training data and unlabeled click data, how

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1086–1095,
Singapore, 6-7 August 2009.

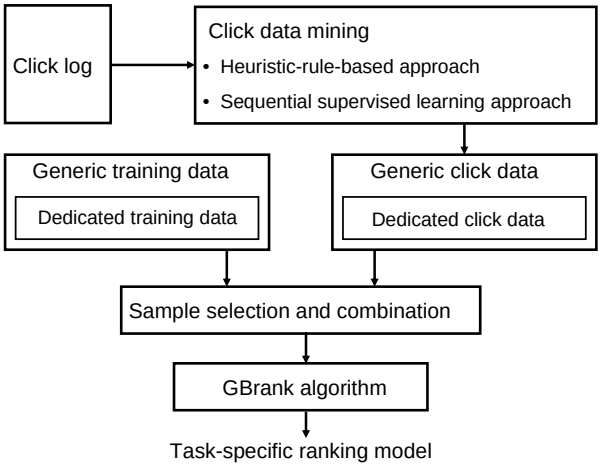


Figure 1: Framework of incorporating click-through data with training data to improve dedicated model for task-specific ranking.

to select and combine them so that the samples have the best utility for learning? As the data distribution for a specific ranking task is different from the generic data distribution, it is natural to select those labeled training samples and unlabeled click preference pairs which belong to this query category, so that the data distributions of training set and testing set are consistent for this category. On the other hand, we should keep in mind that: a) non-dedicated data, i.e, the data that does not belong the specific category, might also have similar distribution as the dedicated data. Such distribution similarity makes non-dedicated data also useful for task-specific rank function learning, especially for the scenario that dedicated training samples is insufficient. b) The quality of dedicated click data may be not as reliable as human labeled training data. In other words, there are some extracted click preference pairs that are inconsistent with human labeling while we regard human labeling as correct labeling.

3) Rank function learning algorithm: we use GBrank (Zheng et al., 2007) algorithm for rank function learning, which has proved to be one of the most effective up-to-date learning-to-rank algorithms; furthermore, GBrank algorithm also takes preference pairs as inputs, which will be illustrated with more details in the paper.

2 Related work

Learning to rank has been a promising research area which continuously improves web search relevance (Burges et al., 2005) (Zha et al., 2006) (Cao et al., 2007) (Freund et al., 1998) (Friedman, 2001) (Joachims, 2002) (Wang and Zhai, 2007) (Zheng et al., 2007). The ranking problem is usually formulated as learning a ranking function from preference data. The basic idea is to minimize the number of contradicted pairs in the training data, and different algorithm cast the preference learning problem from different point of view, for example, RankSVM (Joachims, 2002) uses support vector machines; RankBoost (Freund et al., 1998) applies the idea of boosting from weak learners; GBrank (Zheng et al., 2007) uses gradient boosting with decision tree; RankNet (Burges et al., 2005) uses gradient boosting with neural net-work. In (Zha et al., 2006), query difference is taken into consideration for learning effective retrieval function, which leads to a multi-task learning problem using risk minimization framework.

There are a few related works to apply multiple ranking models for different query categories. However, none of them takes click-through information into consideration. In (Kang and Kim, 2003), queries are categorized into 3 types, informational, navigational and transactional, and different models are applied on each query category.

a KNN method is proposed to employ different ranking models to handle different types of queries (Geng et al., 2008). The KNN method is unsupervised, and it targets to improve the overall ranking instead of the rank-ing for a certain query category. In addition, the KNN method requires all feature vector to be the same.

Quite a few research papers explore how to obtain useful information from click-through data, which could benefit search relevance (Carterette et al., 2008) (Fox et al., 2005) (Radlinski and Joachims, 2007) (Wang and Zhai, 2007). The information can be expressed as pair-wise preferences (Chapelle and Zhang, 2009) (Ji et al., 2009) (Radlinski et al., 2008), or represented as rank features (Agichtein et al., 2006). Task-specific ranking relies on the accuracy of query classification. Query classification or query intention identification has been extensively studied in (Beitzel et al., 2007) (Lee et al., 2005) (Li et al., 2008) (Rose and Levinson, 2004). How to combine editorial data and click data is well discussed in (Chen et al., 2008) (Zheng et al., 2007). In addition, how to use click data to improve ranking are also exploited in personalized or preference-based search (Coyle

Table 1: Statistics of click occurrences for heuristic rule approach.

imp	impression, number of occurrence of the tuple
cc	number of occurrence of the tuple where two documents both get clicked
ncc	number of occurrence of the tuple where url_1 is not clicked but url_2 is clicked
cnc	number of occurrence of the tuple where url_1 is clicked but url_2 is not clicked
$ncnc$	number of occurrence of the tuple where url_1 and url_2 are not clicked

and Smyth, 2007) (Glance, 2001) (R. Jin, 2008).

3 Technical approach

This section presents the related approaches in Figure 1. In Section 4, we will make deeper analysis based on experimental results.

3.1 Click data mining

We use two approaches for click data mining, whose outputs are preference pairs. A *preference pair* is defined as a tuple $\{< x_q, y_q > | x_q \succ y_q\}$, which means for the query q, the document x_q is more relevant than y_q. We need to extract informative and reliable preference pairs which can be used to improve rank function learning.

3.1.1 Heuristic rule approach

We use heuristic rules to extract skip-above pairs and skip-next pairs, which are similar to Strategy 1 (click $>$ skip above) and Strategy 5 (click $>$ no-click next) proposed in (Joachims et al., 2005). To reduce the misleading effect of an individual click behavior, click information from different query sessions is aggregated before applying heuristic rules. For a tuple $(q, \text{url}_1, \text{url}_2, \text{pos}_1, \text{pos}_2)$ where q is query, url_1 and url_2 are urls representing two documents, pos_1 and pos_2 are ranking positions for the two documents with $\text{pos}_1 \prec \text{pos}_2$ meaning url_1 has higher rank than url_2, the statistics for this tuple are listed in Table 1.

Skip-above pair extraction: if ncc is much larger than cnc, and $\frac{cc}{imp}$, $\frac{ncnc}{imp}$ is much smaller than 1, that means, when url_1 is ranked higher than url_2 in query q, most users click url_2 but not click url_1. In this case, we extract a skip-above pair, i.e., url_2 is more relevant than url_1. In order to have highly accurate skip-above pairs, a set of thresholds are applied to only extract the pairs that have high impression and ncc is larger enough than cnc.

Skip-next pair extraction: if $\text{pos}_1 = \text{pos}_2 - 1$, cnc is much larger than ncc, and $\frac{cc}{imp}$, $\frac{ncnc}{imp}$ is much smaller than 1, that means, in most of cases when url_2 is ranked just below url_1 in query q, most users click url_1 but not click url_2. In this case, we regard this tuple as a skip-next pair.

Table 2: Skip-above pairs count vs. human judgements (e.g., the element in the third row and second column means we have 40 skip-above pairs with "excellent" url_1 and "perfect" url_2). P: perfect; E: excellent; G: good; F: fair; B: bad.

	P	E	G	F	B
P	13	13	12	4	0
E	40	44	16	2	2
G	27	53	103	29	8
F	10	15	43	27	5
B	4	4	11	20	14

Table 3: Skip-next pairs vs. human judgements (e.g., the element in the third row and second column means we have 10 skip-next pairs with "excellent" url_1 and "perfect" url_2). P: perfect; E: excellent; G: good; F: fair; B: bad.

	P	E	G	F	B
P	126	343	225	100	35
E	10	71	84	37	12
G	6	9	116	56	21
F	1	5	17	29	14
B	1	1	1	2	5

To test the accuracy of preference pairs, we ask editors to judge some randomly selected pairs from skip-above pairs and skip-next pairs. Editors label each query-url pair using five grades according to relevance: perfect, excellent, good, fair, bad. Table 2 shows skip-above pair distribution. The diagonal elements have high values, which are for tied pairs labeled by editors but determined as skip-above pairs from heuristic rules. Higher values appear in the left-bottom triangle than in the right-top triangle, because there are more skip-above preferences agreed with editors than disagreed with editors. Summing up the tied pairs, agreed and disagreed pairs, 44% skip-above preference judgments agree with editors, 18% skip-above preference judgments disagree with editors,

and there are 38% skip-above pairs judged as tie pairs by editors.

Table 3 shows skip-next pair distribution. Summing up the tied pairs, agreed and disagreed pairs, 70% skip-next preference judgments agree with editors, 4% skip-next preference judgments disagree with editors, and 26% skip-next pairs judged as tie pairs by editors.

Therefore, skip-next pairs have much higher accuracy than skip-above. That is because in a search engine that already has a good ranking function, it is much easier to find a correct skip-next pairs which are consistent with the search engine than to find a correct skip-above pairs which are contradictory to the search engine. Skip-above and skip-next preferences provide us two kinds of users's feedbacks which are complementary: skip-above preferences provide us the feedback that the user's vote is contradictory to the current ranking, which implies the current relative ranking should be reversed; skip-next preferences shows that the user's vote is consistent with the current ranking, which implies the current relative ranking should be maintained with high confidence provided by users' vote.

3.1.2 Sequential supervised learning

The click modeling by sequential supervised learning (SSL) was proposed in (Ji et al., 2009), in which user's sequential click information is exploited to extract relevance information from click-logs. This approach is reliable because 1) the sequential click information embedded in an aggregation of user clicks provides substantial relevance information of the documents displayed in the search results, and 2) the SSL is supervised learning (i.e., human judgments are provided with relevance labels for the training).

The SSL is formulated in the framework of global ranking (Qin et al., 2008). Let $\mathbf{x}^{(q)} = \{x_1^{(q)}, x_2^{(q)}, \ldots, x_n^{(q)}\}$ represent the documents retrieved with a query q, and $\mathbf{y}^{(q)} = \{y_1^{(q)}, y_2^{(q)}, \ldots, y_n^{(q)}\}$ represent the relevance labels assigned to the documents. Here n is the number of documents retrieved with q. Without loss of generality, we assume that n is fixed and invariant with respect to different queries. The SSL determines to find a function F in the form of $\mathbf{y}^{(q)} = F(\mathbf{x}^{(q)})$ that takes all the documents as its inputs, exploiting both local and global information among the documents, and predict the relevance labels of all the document *jointly*. This is distinct to most of learning to rank methods that optimize a ranking model defined on a single document, i.e., in the form of $y_i^{(q)} = f(x_i^{(q)})$, $\forall\, i = 1, 2, \ldots, n$. This formulation of the SSL is important in extracting relevance information from user click data since users' click decisions among different documents displayed in a search session tend to rely not only on the relevance judgment of a single document, but also on the *relative* relevance comparison among the documents displayed; and the global ranking framework is well-formulated to exploit both local and global information from an aggregation of user clicks.

The SSL aggregates all the user sessions for the same query into a tuple <query, n-document list, and an aggregation of user clicks>. Figure 2 illustrates the process of feature extraction from an aggregated session, where $\mathbf{x}^{(q)} = \{x_1^{(q)}, x_2^{(q)}, \ldots, x_n^{(q)}\}$ denotes a sequence of feature vectors extracted from the aggregated session, with $x_i^{(q)}$ representing the feature vector extracted for document i. Specifically, to form feature vector $x_i^{(q)}$, first a feature vector $x_{i,j}^{(q)}$ is extracted from each user j's click information, and $j \in \{1, 2, \ldots\}$, then $x_i^{(q)}$ is formed by averaging over $x_{i,j}^{(q)}$, $\forall j \in \{1, 2, \ldots\}$, i.e., $x_i^{(q)}$ is actually an aggregated feature vector for document i. Table 4 lists all the features used in the SSL modeling. Note that some features are statistics independent of temporal information of the clicks, such as "Position" and "Frequency", while other features reply on their surrounding documents and the click sequences. We use 90,000 query-url pairs to train the SSL model, and 10,000 query-url pairs for best model selection.

With the sequential click modeling discussed above, several sequential supervised algorithms, including the conditional random fields (CRF) (Lafferty et al., 2001), the sliding window method and the recurrent sliding window method (Dietterich, 2002), are explored to find a global ranking function F. We omit the details but refer one to (Ji et al., 2009). The emphasis here is on the importance to adapt these algorithms to the ranking problem.

After training, the SSL model can be used to predict the relevance labels of all the documents in a new aggregated session, and thus pair-wise preference data can be extracted, with the score difference representing the confidence of preference

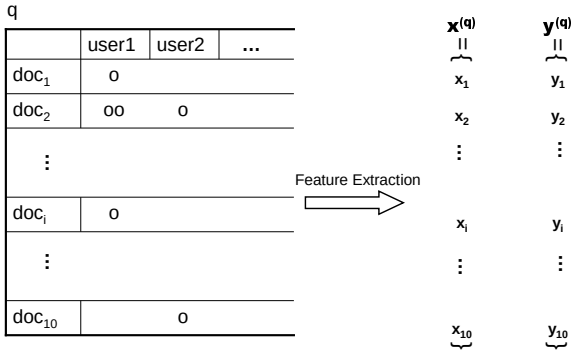


Figure 2: An illustration of feature extraction for an aggregated session for SSL approach. $\mathbf{x}^{(q)}$ denotes an extracted sequence of feature vectors, and $\mathbf{y}^{(q)}$ denotes the corresponding label sequence that is assigned by human judges for training.

Table 4: Click features used in SSL model.

Position	Position of the document in the result list
ClickRank	Rank of 1st click of doc. in click seq.
Frequency	Average number of clicks for this doc.
FrequencyRank	Rank in the list sorted by num. of clicks
IsNextClicked	1 if next position is clicked, 0 otherwise
IsPreClicked	1 if previous position is clicked, 0 otherwise
IsAboveClicked	1 if there is a click above, 0 otherwise
IsBelowClicked	1 if there is a click below, 0 otherwise
ClickDuration	Time spent on the document

prediction. For the reason of convenience, we also call the preference pairs contradicting with production ranking as skip-above pairs and those consistent with production ranking as skip-next pairs, so that we can analyze these two types of preference pairs respectively.

3.2 Modeling algorithm

The basic idea of GBrank (Zheng et al., 2007) is that if the ordering of a preference pair by the ranking function is contradictory to this preference, we need to modify the ranking function along the direction by swapping this prefence pair. Preferences pairs could be generated from labeled data, or could be extracted from click data. For each preference pair $< x, y >$ in the available preference set $S = \{< x_i, y_i > | x_i \succ y_i, i = 1, 2, ..., N\}$, x should be ranked higher than y. In GBrank algorithm, the problem of learning ranking functions is to compute a ranking function h , so that h matches the set of preference, i.e, $h(x_i) \geq h(y_i)$, if $x \succ y$, $i = 1, 2, ..., N$ as many as possible. The following loss function is used to measure the risk of a given ranking function h.

$$R(h) = \frac{1}{2}\sum_{i=1}^{N}(\max\{0, h(y_i) - h(x_i) + \tau\})^2, \quad (1)$$

where τ is the margin between the two documents in the pair. To minimize the loss function, $h(x)$ has to be larger than $h(y)$ with the margin τ, which can be chosen as constant value, or as dynamic values varying with pairs. When pair-wise judgments are extracted from editors' labels with different grades, pair-wise judgments can include grade difference, which can further be used as margin τ. The GBrank algorithm is illustrated in Algorithm 1, and two parameters need to be determined: the shrinkage factor η and the number of iteration.

Algorithm 1 GBrank algorithm.

Start with an initial guess h_0, for $m = 1, 2, ...$

1. Construct a training set: for each $< x_i, y_i > \in S$, derive $(x_i, \max\{0, h_{m-1}(y_i) - h_{m_1}(x_i) + \tau\})$, and
$(y_i, -\max\{0, h_{m-1}(y_i) - h_{m_1}(x_i) + \tau\})$.
2. Fit h_m by using a base regressor with the above training set.
3. $h_m = h_{m-1} + \eta s_m h_m(x)$, where s_m is found by line search to minimize the object function, η is shrinkage factor.

3.3 Sample selection and combination

We use a straightforward approach to learn ranking model from the combined data, which is illustrated in Algorithm 2.

Algorithm 2 Learn ranking model by combining editorial data and click preference pairs.

Input:
- Editorial absolute judgement data.
- Preference pairs from click data.

1. Extract preference pairs from labeled data with absolute judgement.
2. Select and combine preference pairs from click data and labeled data.
3. Learn GBrank model from the combined preference pairs.

Absolute judgement on labeled data contains (query, url) pairs with absolute grade values labeled by human. In Step 1, for each query with

n_q query-url pairs with corresponding grades, $\{<$ query, $\text{url}_i, \text{grade}_i > | i = 1, 2, \ldots, n_q\}$, its preference pairs are extracted as

$\{<$ query, $\text{url}_i, \text{url}_j, \text{grade}_i - \text{grade}_j > | i, j = 1, 2, \ldots, n_q, i \neq j\}$.

When combining human-labeled pairs and click preference pairs, we can give use different relative weights for these two data sources. The loss function becomes

$$R(h) = \frac{w}{N_l} \sum_{i \in \textbf{Labeled}} (\max\{0, h(y_i) - h(x_i) + \tau\})^2 \quad \frac{1-w}{N_c} \sum_{i \in \textbf{Click}} (\max\{0, h(y_i) - h(x_i) + \tau\})^2, \quad (2)$$

where w is used to control the relative weights between labeled training data and click data, N_l is the number of training data pairs, and N_c is the number of click pairs. The margin τ can be determined as grade difference for editor pairs, and be a constant parameter for click pairs.

Step 2 is critical for the efficacy of the approach. A few factors need to be considered:

1) data distribution: for the application of task-specific ranking, our purpose is to improve ranking for the queries belonging to this category. An important observation is that the relevance patterns for the ranking within a specific category may have some unique characteristics, which are different from generic relevance ranking. Thus, it is reasonable to consider only using dedicated labeled training data and dedicated click preference data for training. The reality is that dedicated training data is usually insufficient, while it is possible that non-dedicated data can also help the learning.

2) click pair quality: it is inevitable there exist some incorrect pairs in the click preference pairs. Such incorrect pairs may mislead the learning. So overall, can the click preference pairs still help the learning for task-specific ranking? By our study, skip-above pairs usually contain more incorrect pairs compared with skip-above pairs. Does this mean skip-next pairs are always more helpful in improving learning than skip-above pairs?

3) click pair utility: use labeled training data as baseline, how much complimentary information can click pairs bring? This is determined by the methodology of click data mining approach.

While it is possible to achieve some learning improvement for task-specific ranking by using click pairs by a plausible method, we attempt to empirically explore the above interweaving factors for deeper understanding, in order to apply the most appropriate strategy to exploit click data on real-world applications of task-specific ranking.

4 Experiments

4.1 Data set

Query category: in the experiments, we use long query ranking as an example of task-specific ranking, because it is commonly known that long query ranking has some unique relevance patterns compared with generic ranking. We define the long queries as the queries containing at least three tokens. The techniques and analysis proposed in this paper can be applied to other ranking tasks, such as rankings for specific query segments like time-sensitive queries, navigational queries, or rankings for specific domains/contents like answers, blogs, news, as long as the tasks have their own characteristics of data distributions and discriminant rank features.

Labeled training data: we do experiments based on a data set for a commercial search engine, for which there are 16,797 query-url pairs (with 1,123 different queries) that have been labeled by editors. The proportion of long queries is about 35% of all queries. The data distribution of such long queries may be different from general data distribution, as it will be validated in the experiments below.

The human labeled data is randomly split into two sets: training set (8,831 query-url pairs, 589 queries), and testing set (7,966 query-url pairs, 534 queries). The training set will be combined with click preference pairs for rank function learning, and the testing set will be used to evaluate the efficacy of the ranking function. In the training set, there are 3,842 long query-url pairs (229 queries). At testing stage, the learned rank functions are applied only to the long queries in the testing data, as our concern in this paper is how to improve task-specific ranking, i.e., long query ranking in the experiment. In the testing data, there are 3,210 query-url pairs (193 queries) are long query data, which will be used to test rank functions.

Click preference pairs: using the two approaches of heuristic rule approach and sequential supervised approach, we extract click prefence pairs from the click log of the search engine. Each approach yields both skip-next and skip-above pairs, which are sorted by confidence descending order respectively.

Table 5: Use click data by heuristic rule approach (Data Selection: "N": not use; "D": use dedicated data; "G": use generic data. Data Source: "T": training data; "C": click data)

(a) skip-next pairs

	NT	DT	GT
NC	n/a	0.7736	0.7813
DC	0.7822	0.7906 (1.2%)	0.7997(2.4%)
GC	0.7834	0.7908 (1.2%)	0.7950 (1.7%)

(b) skip-above pairs

	NT	DT	GT
NC	n/a	0.7736	0.7813
DC	0.6649	0.7676 (-1.6%)	0.7748 (-0.8%)
GC	0.6792	0.7656 (-2.0%)	0.7989 (2.2%)

Table 6: Use click data by SSL approach (Data Selection: "N": not use; "D": use dedicated data; "G": use generic data. Data Source: "T": training data; "C": click data)

(a) skip-next pairs

	NT	DT	GT
NC	n/a	0.7736	0.7813
DC	0.7752	0.7933 (1.5%)	0.7936 (1.5%)
GC	0.7624	0.7844 (0.4%)	0.7914 (1.2%)

(b) skip-above pairs

	NT	DT	GT
NC	n/a	0.7736	0.7813
DC	0.6756	0.7636 (-2.2%)	0.7784 (-0.3%)
GC	0.6860	0.7717 (-1.2%)	0.7774 (-0.5%)

4.2 Setup and measurements

We try different sample selection and combination strategies to train rank functions using GBrank algorithm. For the labeled training data, we either use generic data or dedicated data. For the click preference pairs, we also try these two options. Furthermore, as more click preference pairs may bring more useful information to help the learning while on the other hand, the more incorrect pairs may be given so that they mislead the learning, we try different amounts of these prefence pairs: 5,000, 10,000, 30,000, 50,000, 70,000 and 100,000 pairs.

We use NDCG to evaluate ranking model, which is defined as

$$\text{NDCG}_n = Z_n \sum_{i=1}^{n} \frac{2^{r(i)}-1}{\log(i+1)}$$

where i is the position in the document list, $r(i)$ is the score of Document i, and Z_n is a normalization factor, which is used to make the NDCG of ideal list be 1.

4.3 Results

Table 5 and 6 show the NDCG_5 results by using heuristic rule approach and SSL approach respectively. We do not present NDCG_1 results due to space limitation, but NDCG_1 results have the similar trends as NDCG_5.

Baseline by training data: there are two baseline functions by using training data sets 1) use dedicated training data (DT), NDCG_5 on the testing set by the rank function is 0.7736; 2) use generic training data (GT), NDCG_5 is 0.7813. It is reasonable that using generic training data is better than only using dedicated training data, because the distributions of non-dedicated data and dedicated data share some similarity. As the dedicated training data is insufficient, the adoption of the extra non-dedicated data helps the learning. We compare learning results with Baseline 2) (use generic training data, the slot of NC + GT in the tables), which is the higher baseline.

Baseline by click data: we then study the utilities of click preference pairs by using them alone for training without using labeled training data. In Table 5 and 6, each of the NDCG_5 results using click preference pairs is the highest NDCG_5 value over the cases of using different amounts of pairs (5000, 10,000, 30,000, 50,000, 70,000 and 100,000 pairs). The results regarding the pairs amounts are illustrated in Figure 3, which will help us to analyze the results more deeply.

If we only use click preference pairs for training (the two table slots DC+NT and GC+NT, corresponding to using dedicated click preference pairs and generic click pairs respectively), the best case is using skip-next pairs extracted by heuristic rule approach (Table 5 (a)). It is not surprising that skip-next pairs outperform skip-above pairs because there are significantly lower percentage of incorrect pairs in skip-next pairs compared with skip-above pairs. It is a little bit surprising that the case of DC+NT has no dominant advantage over GC+NT as we expected. For example, in Table 5 (a), the NDCG_5 values (0.7822 and 0.7834) are very close to each other. However, in Figure 3, we find that with the same amount of pairs, when we use 30,000 or fewer pairs, using dedi-

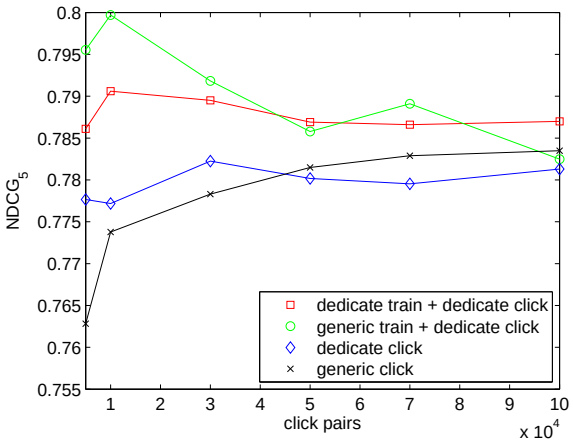


Figure 3: Incorporate different amounts of skip-next pairs by heuristic rule approach with generic training data.

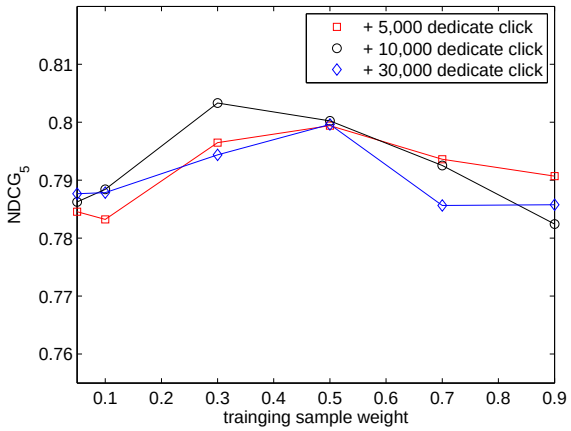


Figure 4: The effects of using different combining weights. Skip-next pairs by heuristic rule approach are combined with generic training data.

cated click pairs alone is always better than using generic click pairs alone. With more click pairs being used ($> 30,000$), the noise rates become higher in the pairs, which makes the distribution factor less important.

Combine training data and click data: we compare the four table slots, DC+DT, GC+DT, DC+GT, GC+GT, in Table 5 and 6, and there are quite a few interesting observations:

1) *Skip-next vs. skip-above*: overall, incorporating skip-next pairs with training data is better than incorporating skip-above pairs, due to the reason that there are more incorrect pairs in skip-above pairs, which may mislead the learning. The only exception is the slot GC+GT in Table 5 (b), whose NDCG_5 improvement is as high as 2.2%. We further track this result, and find that this is the case by using only 5,000 generic skip-above pairs. The noise rate of these 5,000 pairs is low because they have the highest pair extraction confidence values. At the same time, these 5,000 pairs may provide good complementary signals to the generic training data, so that the learning result is good. However, in general, skip-next pairs have better utilities than skip-above pairs.

2) *Dedicated training data vs. generic training data*: using generic training data is generally better than only using dedicated training data. If training data is insufficient, the extra non-dedicated data provides useful information for relevance pattern learning, and the distribution dissimilarity between dedicated data and non-dedicated data is not the most important factor.

3) *Dedicated click data vs. generic click data*: using dedicated click data is more effective than using generic click data. From Figure 3, we observe that when 30,000 or fewer pairs are incorporated into training data, using dedicate click pairs is always better than using generic click pairs.

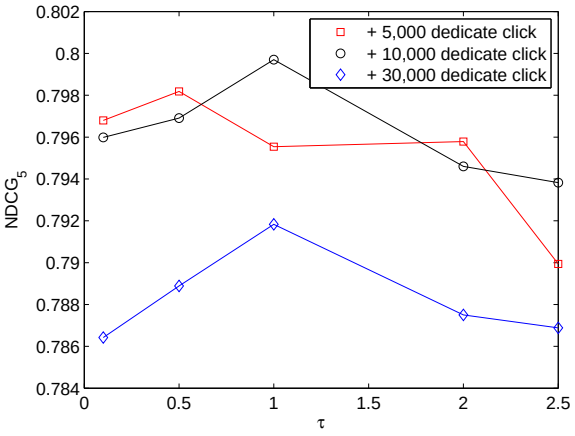


Figure 5: The effects of using different margin values for click preference pairs. Skip-next pairs by heuristic rule approach are incorporated with generic training data.

4) *Heuristic rule approach vs. SSL approach*: the preference pairs extracted by heuristic rule approach have better utilities than those extracted by SSL approach.

5) *GBrank parameters for combining training data and click pairs*: the relative weight w for combining training data and click pairs in (2) may also affect rank function learning. Figure 4 shows the effects of using different combining weights,

for which skip-next pairs by heuristic rule approach are combined with generic training data. We observe that neither over-weighting training data or over-weighting click pairs yields good results while the two data sources are best exploited at certain weight values when there is good balance between them. Another concern is the appropriate margin value τ for the click pairs in (2). Figure 5 shows that $\tau = 1$ consistently yields good learning results, which suggests us that click pair provides good information at $\tau = 1$.

4.4 Discussions

we have defactorized the related approaches for exploiting click data to improve task-specific rank learning. The utility of click preference pairs depends on the following factors:

1) Data distribution: if click pairs have good quality, we should use dedicated click pairs instead of generic click pairs, so that the samples for training have similar distribution to the task of task-specific ranking.

2) The amount of dedicated training data: the more dedicated training data, the more reliable the task-specific rank function is; thus, the less room for learning improvement using click data. For the case in the experiment that dedicated training is insufficient, the non-dedicated training data can also help the learning as non-dedicated training data share relevance pattern similarity with the dedicated data distribution.

3) The quality of click pairs: if we can extract large amount of high-quality click pairs, the learning improvement will be significant. For example, as shown in Figure 3, at the early stage with fewer click pairs (5,000 and 10,000 pairs) being combined with training data, the learning improvement is best. With more click pairs are used, the noise rate in the click pairs becomes higher so that the learning misleading factor is more important than information complementary factor. Thus, it is important to improve the reliability of the click pairs.

4) The utility of click pairs: by our study, the quality of click pairs extracted by SSL approach is comparable to those extracted by heuristic rule approach. The possible reason that heuristic-rule-based click pairs can bring more benefit is that these pairs provide more complementary information compared with SSL approach. As the methodologies of these two click data extraction approaches are totally different, in future we will explore the concrete reason that causes such utility difference.

5 Conclusions

By empirically exploring the related factors in utilizing click-through data to improve dedicated model learning for task-specific ranking, we have better understood the principles of using click preference pairs appropriately, which is important for the real-world applications in commercial search engines as using click data can significantly save human labeling costs and makes rank function learning more efficient. In the case that dedicated training data is limited, while non-dedicated training data is helpful, using dedicated skip-next pairs is the most effective way to further improve the learning. Heuristic rule approach provides more useful click pairs compared with sequential supervised learning approach. The quality of click pairs is critical for the efficacy of the approach. Therefore, an interesting topic is how to further reduce the inconsistency between skip-above pairs and human labeling so that such data may also be useful for task-specific ranking.

References

E. Agichtein, E. Brill, and S. Dumais. 2006. Improving web search ranking by incorporating user behavior information. *Proc. of ACM SIGIR Conference*.

S. M. Beitzel, E. C. Jensen, A. Chowdhury, and O. Frieder. 2007. Varying approaches to topical web query classification. *Proceedings of ACM SIGIR conference*.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. 2005. Learning to rank using gradient descent. *Proc. of Intl. Conf. on Machine Learning*.

Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li. 2007. Learning to rank: From pairwise approach to listwise. *Proceedings of ICML conference*.

B. Carterette, P. N. Bennett, D. M. Chickering, and S. T. Dumais. 2008. Here or there: preference judgments for relevance. *Proc. of ECIR*.

O. Chapelle and Y. Zhang. 2009. A dynamic bayesian network click model for web search ranking. *Proceedings of the 18th International World Wide Web Conference*.

K. Chen, Y. Zhang, Z. Zheng, H. Zha, and G. Sun. 2008. Adapting ranking functions to user preference. *ICDE Workshops*, pages 580–587.

M. Coyle and B. Smyth. 2007. Supporting intelligent web search. *ACM Transaction Internet Tech.*, 7(4).

T. G. Dietterich. 2002. Machine learning for sequential data: a review. *Lecture Notes in Computer Science*, (2396):15–30.

S. Fox, K. Karnawat, M. Mydland, S. Dumias, and T. White. 2005. Evaluating implicit measures to improve web search. *ACM Trans. on Information Systems*, 23(2):147–168.

Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. 1998. An efficient boosting algorithm for combining preferences. *Proceedings of International Conference on Machine Learning*.

J. Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Ann. Statist.*, 29:1189–1232.

X. Geng, T. Liu, T. Qin, A. Arnold, H. Li, and H. Shum. 2008. Query dependent ranking with k nearest neighbor. *Proceedings of ACM SIGIR Conference*.

N. S. Glance. 2001. Community search assistant. *Intelligent User Interfaces*, pages 91–96.

S. Ji, K. Zhou, C. Liao, Z. Zheng, G. Xue, O. Chapelle, G. Sun, and H. Zha. 2009. Global ranking by exploiting user clicks. In *SIGIR'09*, Boston, USA, July 19-23.

T. Joachims, L. Granka, B. Pan, and G Gay. 2005. Accurately interpreting clickthough data as implicit feedback. *Proc. of ACM SIGIR Conference*.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

I. Kang and G. Kim. 2003. Query type classification for web document retrieval. *Proceedings of ACM SIGIR Conference*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

U. Lee, Z. Liu, and J. Cho. 2005. Automatic identification of user goals in web search. *Proceedings of International Conference on World Wide Web*.

X. Li, Y.Y Wang, and A. Acero. 2008. Learning query intent from regularized click graphs. *Proceedings of ACM SIGIR Conference*.

T. Y Liu. 2008. Learning to rank for information retrieval. *SIGIR tutorial*.

T. Qin, T. Liu, X. Zhang, D. Wang, and H. Li. 2008. Global ranking using continuous conditional random fields. In *NIPS*.

H. Li R. Jin, H. Valizadegan. 2008. Ranking refinement and its application to information retrieval. *Proceedings of International Conference on World Wide Web*.

F. Radlinski and T. Joachims. 2007. Active exploration for learning rankings from clickthrough data. *Proc. of ACM SIGKDD Conference*.

F. Radlinski, M. Kurup, and T. Joachims. 2008. How does clickthrough data reflect retrieval quality? *Proceedings of ACM CIKM Conference*.

D. E. Rose and D. Levinson. 2004. Understanding user goals in web search. *Proceedings of International Conference on World Wide Web*.

X. Wang and C. Zhai. 2007. Learn from web search logs to organize search results. In *Proceedings of the 30th ACM SIGIR*.

H. Zha, Z. Zheng, H. Fu, and G. Sun. 2006. Incorporating query difference for learning retrieval functions in world wide web search. *Proceedings of the 15th ACM Conference on Information and Knowledge Management*.

Z. Zheng, H. Zhang, T. Zhang, O. Chapelle, K. Chen, and G. Sun. 2007. A general boosting method and its application to learning ranking functions for web search. *NIPS*.

The Feature Subspace Method for SMT System Combination

Nan Duan[1], Mu Li[2], Tong Xiao[3], Ming Zhou[2]
[1]Tianjin University, Tianjin, China
[2]Microsoft Research Asia, Beijing, China
[3]Northeastern University, Shenyang, China
{v-naduan,muli,v-toxiao,mingzhou}@microsoft.com

Abstract

Recently system combination has been shown to be an effective way to improve translation quality over single machine translation systems. In this paper, we present a simple and effective method to systematically derive an ensemble of SMT systems from one baseline linear SMT model for use in system combination. Each system in the resulting ensemble is based on a feature set derived from the features of the baseline model (typically a subset of it). We will discuss the principles to determine the feature sets for derived systems, and present in detail the system combination model used in our work. Evaluation is performed on the data sets for NIST 2004 and NIST 2005 Chinese-to-English machine translation tasks. Experimental results show that our method can bring significant improvements to baseline systems with state-of-the-art performance.

1 Introduction

Research on Statistical Machine Translation (SMT) has shown substantial progress in recent years. Since the success of phrase-based methods (Och and Ney, 2004; Koehn, 2004), models based on formal syntax (Chiang, 2005) or linguistic syntax (Liu et al., 2006; Marcu et al., 2006) have also achieved state-of-the-art performance. As a result of the increasing numbers of available machine translation systems, studies on system combination have been drawing more and more attention in SMT research.

There have been many successful attempts to combine outputs from multiple machine translation systems to further improve translation quality. A system combination model usually takes *n*-best translations of single systems as input, and depending on the combination strategy, different methods can be used. Sentence-level combination methods directly select hypotheses from original outputs of single SMT systems (Sim et al., 2007; Hildebrand and Vogel, 2008), while phrase-level or word–level combination methods are more complicated and could produce new translations different from any translations in the input (Bangalore et al., 2001; Jayaraman and Lavie, 2005; Matusov et al., 2006; Sim et al., 2007).

Among all the factors contributing to the success of system combination, there is no doubt that the availability of multiple machine translation systems is an indispensable premise. Although various approaches to SMT system combination have been explored, including enhanced combination model structure (Rosti et al., 2007), better word alignment between translations (Ayan et al., 2008; He et al., 2008) and improved confusion network construction (Rosti et al., 2008), most previous work simply used the ensemble of SMT systems based on different models and paradigms at hand and did not tackle the issue of how to obtain the ensemble in a principled way. To our knowledge the only work discussed this problem is Macherey and Och (2007), in which they experimented with building different SMT systems by varying one or more sub-models (i.e. translation model or distortion model) of an existing SMT system, and observed that changes in early-stage model training introduced most diversities in translation outputs.

In this paper, we address the problem of building an ensemble of diversified machine translation systems from a single translation engine for system combination. In particular, we propose a novel *Feature Subspace* method for the ensemble construction based on any baseline SMT model which can be formulated as a standard linear function. Each system within the ensemble is based on a group of features directly derived from the baseline model with minimal efforts (which is typically a subset of the features used in the baseline model), and the resulting system is optimized in the derived feature space accordingly.

We evaluated our method on the test sets for NIST 2004 and NIST 2005 Chinese-to-English

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1096–1104,
Singapore, 6-7 August 2009.

machine translation tasks using two baseline SMT systems with state-of-the-art performance. Experimental results show that the feature subspace method can bring significant improvements to both baseline systems.

The rest of the paper is organized as follows. The motivation of our work is described on Section 2. In Section 3, we first give a detailed description about feature subspace method, including the principle to select subspaces from all possible options, and then an *n*-gram consensus – based sentence-level system combination method is presented. Experimental results are given in Section 4. Section 5 discusses some related issues and concludes the paper.

2 Motivation

Our motivations for this work can be described in the following two aspects.

The first aspect is related to the cost of building single systems for system combination. In previous work, the SMT systems used in combination differ mostly in two ways. One is the underlying models adopted by individual systems. For example, using an ensemble of systems respectively based on phrase-based models, hierarchical models or even syntax-based models is a common practice. The other is the methods used for feature function estimation such as using different word alignment models, language models or distortion models. For the first solution, building a new SMT system with different methodology is by no means an easy task even for an experienced SMT researcher, because it requires not only considerable effects to develop but also plenty of time to accumulate enough experiences to fine tune the system. For the second alternative, usually it requires time-consuming retraining for word alignment or language models. Also some of the feature tweaking in this solution is system or language specific, thus for any new systems or language pairs, human engineering has to be involved. For example, using different word segmentation methods for Chinese can generate different word alignment results, and based on which a new SMT system can be built. Although this may be useful to combination of Chinese-to-English translation, it is not applicable to most of other language pairs. Therefore it will be very helpful if there is a light-weight method that enables the SMT system ensemble to be systematically constructed based on an existing SMT system.

Source sentence	中国 最大 规模 的 海水 淡化 工程 落户 舟山
Ref translation	China's largest sea water desalinization project settles in Zhoushan
Default translation	China 's largest desalination project in Zhoushan
FS_{-PEF} translation	China 's largest sea water desalination project in Zhoushan

Table 1: An example of translations generated from the same decoder but with different feature settings.

	Chinese	English	$p(e\|f)$
1	海水 淡化	desalination	0.4000
2	海水	sea water	0.1748
3	淡化	desalination	0.0923

Table 2: Parameters of related phrases for examples in Table 1.

The second aspect motivating our work comes from the subspace learning method in machine learning literature (Ho, 1998), in which an ensemble of classifiers are trained on subspaces of the full feature space, and final classification results are based on the vote of all classifiers in the ensemble. Lopez and Resnik (2006) also showed that feature engineering could be used to overcome deficiencies of poor alignment. To illustrate the usefulness of feature subspace in the SMT task, we start with the example shown in Table 1. In the example, the Chinese source sentence is translated with two settings of a hierarchical phrase-based system (Chiang, 2005). In the default setting all the features are used as usual in the decoder, and we find that the translation of the Chinese word 海水 (sea water) is missing in the output. This can be explained with the data shown in Table 2. Because of noises and word alignment errors in the parallel training data, the inaccurate translation phrase 海水 淡化 ⇒ $desalination$ is assigned with a high value of the phrase translation probability feature $p(e|f)$. Although the correct translation can also be composed by two phrases 海水 ⇒ $sea\ water$ and 淡化 ⇒ $desalination$, its overall translation score cannot beat the incorrect one because the combined phrase translation probability of these two phrases are much smaller than $p(desalination|$海水 淡化$)$. However, if we intentionally remove the $p(e|f)$ feature from the model, the preferred translation can be generated as shown in the result of FS_{-PEF} because in

this way the bad estimation of $p(e|f)$ for this phrase is avoided.

This example gives us the hint that building decoders based on subspaces of a standard model could help with working around some negative impacts of inaccurate estimations of feature values for some input sentences. The subspace-based systems are expected to work similarly to statistical classifiers trained on subspaces of a full feature space – though the overall accuracy of baseline system might be better than any individual systems, for a specific sentence some individual systems could generate better translations. It is expected that employing an ensemble of subspace-based systems and making use of consensus between them will outperform the baseline system.

3 Feature Subspace Method for SMT System Ensemble Construction

In this section, we will present in detail the method for systematically deriving SMT systems from a standard linear SMT model based on feature subspaces for system combination.

3.1 SMT System Ensemble Generation

Nowadays most of the state-of-the-art SMT systems are based on linear models as proposed in Och and Ney (2002). Let $h_m(f,e)$ be a feature function, and λ_m be its weight, an SMT model D can be formally written as:

$$e^* = \arg\max_e \sum_m \lambda_m h_m(f,e) \quad (1)$$

Noticing that Equation (1) is a general formulation independent of any specific features, technically for any subset of features used in D, a new SMT system can be constructed based on it, which we call a *sub-system*.

Next we will use Ω to denote the full feature space defined by the entire set of features used in D, and $s \subseteq \Omega$ is a feature subset that belongs to $\rho(\Omega)$, the power set of Ω. The derived sub-system based on subset $s \subseteq \Omega$ is denoted by d_s. Although in theory we can use all the sub-systems derived from every feature subset in $\rho(\Omega)$, it is still desirable to use only some of them in practice. The reasons for this are two-fold. First, the number of possible sub-systems ($2^{|\Omega|}$) is exponential to the size of Ω. Even when the number of features in Ω is relatively small, i.e. 10, there will be up to 1024 sub-systems in total, which is a large number for combination task. Larger feature sets will make the system combination practically infeasible. Second, not every sub-system could contribute to the system combination. For example, feature subsets only containing very small number of features will lead to sub-systems with very poor performance; and the language model feature is too important to be ignored for a sub-system to achieve reasonably good performance.

In our work, we only consider feature subspaces with only one difference from the features in Ω. For each non- language model feature h_i, a sub-system d_i is built by removing h_i from Ω. Allowing for the importance of the language model (LM) feature to an SMT model, we do not remove any LM feature from any sub-system. Instead, we try to weaken the strength of a LM feature by lowering its *n*-gram order. For example, if a 4-gram language model is used in the baseline system D, then a trigram model can be used in one sub-system, and a bigram model can be used in another. In this way more than one sub-system can be derived based on one LM feature. When varying a language model feature, the *one-feature difference* principle is still kept: if we lower the order of a language model feature, no other features are removed or changed.

The remaining issue of using weakened LM features is that the resulting ensemble is no longer strictly based on subspace of Ω. However, this theoretical imperfection can be remedied by introducing Ω', a super-space of Ω to include all lower-order LM features. In this way, an augmented baseline system D' can be built based on Ω', and the baseline system D itself can also be viewed as a sub-system of D'. We will show in the experimental section that D' actually performs even slightly better than the original baseline system D, but results of sub-system combination are significantly better that both D and D'.

After the sub-system ensemble is constructed, each sub-system tunes its feature weights independently to optimize the evaluation metrics on the development set.

Let $\mathcal{D} = \{d_1, \ldots, d_n\}$ be the set of sub-systems obtained by either removing one non-LM feature or changing the order of a LM feature, and $\mathcal{H}_i$ be the *n*-best list produced by d_i. Then $\mathcal{H}(\mathcal{D})$, the *translation candidate pool* to the system combination model can be written as:

$$\mathcal{H}(\mathcal{D}) = \bigcup_i \mathcal{H}_i \quad (2)$$

The advantage of this method is that it allows us to systematically build an ensemble of SMT systems at a very low cost. From the decoding

perspective, all the sub-systems share a common decoder, with minimal extensions to the baseline systems to support the use of specified subset of feature functions to compute the overall score for translation hypotheses. From the model training perspective, all the non-LM feature functions can be estimated once for all sub-systems. The only exception is the language model feature, which may be of different values across multiple sub-systems. However, since lower-order models have already been contained in higher-order model for the purpose of smoothing in almost all statistical language model implementations, there is also no extra training cost.

3.2 System Combination Scheme

In our work, we use a sentence-level system combination model to select best translation hypothesis from the candidate pool $\mathcal{H}(\mathcal{D})$. This method can also be viewed to be a hypotheses re-ranking model since we only use the existing translations instead of performing decoding over a confusion network as done in the word-level combination method (Rosti et al., 2007).

The score function in our combination model is formulated as follows:

$$e^* = \underset{e \in \mathcal{H}(\mathcal{D})}{argmax} \lambda_{LM} h_{LM}(e) + \lambda_l L + \psi(e, \mathcal{H}(\mathcal{D})) \tag{3}$$

where $h_{LM}(e)$ is the language model score for e, L is the length of e, and $\psi(e, \mathcal{H}(\mathcal{D}))$ is a translation consensus –based scoring function. The computation of $\psi(e, \mathcal{H}(\mathcal{D}))$ is further decomposed into weighted linear combination of a set of *n*-gram consensus –based features, which are defined in terms of the order of *n*-gram to be matched between current candidate and other translation in $\mathcal{H}(\mathcal{D})$.

Given a translation candidate e, the *n*-gram agreement feature between e and other translations in the candidate pool is defined as:

$$h_n^+(e, \mathcal{H}(\mathcal{D})) = \sum_{e' \in \mathcal{H}(\mathcal{D}), e' \neq e} G_n(e, e') \tag{4}$$

where the function $G_n(e, e')$ counts the occurrences of *n*-grams of e in e':

$$G_n(e, e') = \sum_{i=1}^{|e|-n+1} \delta(e_i^{i+n-1}, e') \tag{5}$$

Here $\delta(\cdot,\cdot)$ is the indicator function - $\delta(e_i^{i+n-1}, e')$ is 1 when the *n*-gram e_i^{i+n-1} appears in e', otherwise it is 0.

In order to give the combination model an opportunity to penalize long but inaccurate translations, we also introduce a set of *n*-gram disagreement features in the combination model:

$$h_n^-(e, \mathcal{H}(\mathcal{D})) = \sum_{e' \in \mathcal{H}(\mathcal{D}), e' \neq e} (|e| - n + 1 - G_n(e, e')) \tag{6}$$

Because each order of *n*-gram match introduces two features, the total number of features in the combination model will be $2m + 2$ if m orders of *n*-gram are to be matched in computing $\psi(e, \mathcal{H}(\mathcal{D}))$. Since we also adopt a linear scoring function in Equation (3), the feature weights of our combination model can also be tuned on a development data set to optimize the specified evaluation metrics using the standard Minimum Error Rate Training (MERT) algorithm (Och 2003).

Our method is similar to the work proposed by Hildebrand and Vogel (2008). However, except the language model and translation length, we only use intra-hypothesis *n*-gram agreement features as Hildebrand and Vogel did and use additional intra-hypothesis *n*-gram disagreement features as Li et al. (2009) did in their co-decoding method.

4 Experiments

4.1 Data

Experiments were conducted on the NIST evaluation sets of 2004 (MT04) and 2005 (MT05) for Chinese-to-English translation tasks. Both corpora provide 4 reference translations per source sentence. Parameters were tuned with MERT algorithm (Och, 2003) on the NIST evaluation set of 2003 (MT03) for both the baseline systems and the system combination model. Translation performance was measured in terms of case-insensitive NIST version of BLEU score which computes the brevity penalty using the shortest reference translation for each segment, and all the results will be reported in percentage numbers. Statistical significance is computed using the bootstrap re-sampling method proposed by Koehn (2004). Statistics of the data sets are summarized in Table 3.

Data set	#Sentences	#Words
MT03 (dev)	919	23,782
MT04 (test)	1,788	47,762
MT05 (test)	1,082	29,258

Table 3: Data set statistics.

We use the parallel data available for the NIST 2008 constrained track of Chinese-to-English machine translation task as bilingual training data, which contains 5.1M sentence pairs, 128M Chinese words and 147M English words after pre-processing. GIZA++ toolkit (Och and Ney, 2003) is used to perform word alignment in both directions with default settings, and the intersect-diag-grow method is used to generate symmetric word alignment refinement. The language model used for all systems is a 5-gram model trained with the English part of bilingual data and Xinhua portion of LDC English Gigaword corpus version 3. In experiments, multiple language model features with the order ranging from 2 to 5 can be easily obtained from the 5-gram one without retraining.

4.2 System Description

Theoretically our method is applicable to all linear model –based SMT systems. In our experiments, two in-house developed systems are used to validate our method. The first one (**SYS1**) is a system based on the hierarchical phrase-based model as proposed in (Chiang, 2005). Phrasal rules are extracted from all bilingual sentence pairs, while hierarchical rules with variables are extracted from selected data sets including LDC2003E14, LDC2003E07, LDC2005T06 and LDC2005T10, which contain around 350,000 sentence pairs, 8.8M Chinese words and 10.3M English words. The second one (**SYS2**) is a re-implementation of a phrase-based decoder with lexicalized reordering model based on maximum entropy principle proposed by Xiong et al. (2006). All bilingual data are used to extract phrases up to length 3 on the source side.

In following experiments, we only consider removing common features shared by both baseline systems for feature subspace generation. Rule penalty feature and lexicalized reordering feature, which are particular to SYS1 and SYS2, are not used. We list the features in consideration as follows:

- ***PEF*** and ***PFE***: phrase translation probabilities $p(e|f)$ and $p(f|e)$
- ***PEFLEX*** and ***PFELEX***: lexical weights $p_{lex}(e|f)$ and $p_{lex}(f|e)$
- ***PP***: phrase penalty
- ***WP***: word penalty
- ***BLP***: bi-lexicon pair counting how many entries of a conventional lexicon co-occurring in a given translation pair
- ***LM-n***: language model with order *n*

Based on the principle described in Section 3.1, we generate a number of feature subspaces for each baseline system as follows:

- *For non-LM features* (*PEF, PFE, PEFLEX, PFELEX, PP*, *WP* and *BLP*), we remove one of them from the full feature space each time. Thus 7 feature subspaces are generated, which are denoted as FS_{-PEF}, FS_{-PFE}, $FS_{-PEFLEX}$, $FS_{-PFELEX}$, FS_{-PP}, FS_{-WP} and FS_{-BLP} respectively. The 5-gram LM feature is used in each of them.
- *For LM features* (*LM-n*), we change the order from 2 to 5 with all the other non-LM features present. Thus 4 LM-related feature subspaces are generated, which are denoted as FS_{LM-2}, FS_{LM-3}, FS_{LM-4} and FS_{LM-5} respectively. FS_{LM-5} is essentially the full feature space of baseline system.

For each baseline system, we construct a total of 11 sub-systems by using above feature subspaces. The baseline system is also contained within them because of using FS_{LM-5}. We call all sub-systems are *non-baseline sub-systems* except the one derived by using FS_{LM-5}.

By default, the beam size of 60 is used for all systems in our experiments. The size of *n*-best list is set to 20 for each sub-system, and for baseline systems, this size is set to 220, which equals to the size of the combined *n*-best list generated by total 11 sub-systems. The order of *n*-gram agreement and disagreement features used in sentence-level combination model ranges from unigram to 4-gram.

4.3 Evaluation of Oracle Translations

We first evaluate the oracle performance on the *n*-best lists of baseline systems and on the combined *n*-best lists of sub-systems generated from each baseline system.

The oracle translations are obtained by using the metric of sentence-level BLEU score (Ye et al., 2007). Table 4 shows the evaluation results, in which ***Baseline*** stands for baseline system with a 5-gram LM feature, and ***FS*** stands for 11 sub-systems derived from the baseline system.

		SYS1	SYS2
		BLEU/TER	BLEU/TER
MT04	*Baseline*	49.68/0.6411	49.50/0.6349
	FS	**51.05/0.6089**	**50.53/0.6056**
MT05	*Baseline*	48.89/0.5946	48.37/0.5944
	FS	**50.69/0.5695**	**49.81/0.5684**

Table 4: Oracle BLEU and TER scores on baseline systems and their generated sub-systems.

For both SYS1 and SYS2, feature subspace method achieves higher oracle BLEU and lower TER scores on both MT04 and MT05 test sets, which gives the feature subspace method more potential to achieve higher performance than the baseline systems.

We then investigate the ratio of translation candidates in the combined *n*-best lists of non-baseline sub-systems that are not included in the baseline's *n*-best list. Table 5 shows the statistics.

	MT04	MT05
SYS1	69.71%	69.69%
SYS2	59.07%	58.54%

Table 5: Ratio of unique translation candidates from non-baseline sub-systems.

From Table 5 we can see that only less than half of the translation candidates of sub-systems overlap with those the of baseline systems. This result, together with the oracle BLEU and TER score estimation, helps eliminate the concern that no diversities or better translation candidates can be obtained by using sub-systems.

4.4 Feature Subspace Method on Single SMT System

Next we validate the effect of feature subspace method on single SMT systems.

Figure 1 shows the evaluation results of different systems on the MT05 test set. From the figure we can see that the overall accuracy of baseline systems is better than any of their derived sub-systems, and except the sub-system derived by using FS_{LM-2}, the performance of all the systems are fairly similar.

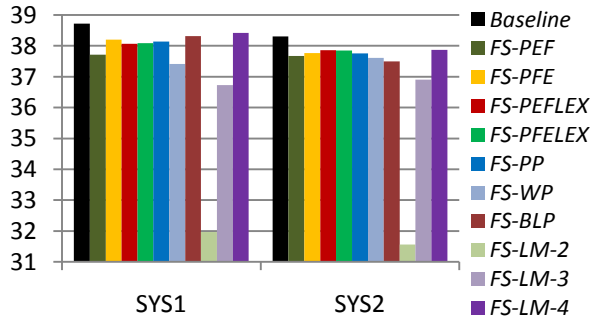


Figure 1: Performances of different systems.

We then evaluate the system combination method proposed in Section 3.2 with all the sub-systems for each baseline system. Table 6 shows the results on both MT04 and MT05 data sets, in which ***FS-Comb*** denotes the system combination using 11 sub-systems.

From Table 6 we can see that by using *FS-Comb* we obtain about 1.1~1.3 points of BLEU gains over baseline systems. We also include in Table 6 the results for ***Baseline+mLM***, which stands for the augmented baseline system as described in Section 3.1 using a bunch of LM features from bigram to 5-gram. It can be seen that both augmented baseline systems outperform their corresponding baseline systems slightly but consistently on both data sets.

		MT04	MT05
SYS1	*Baseline*	39.07	38.72
	Baseline+mLM	39.34+	39.14+
	FS-Comb	**40.43**++	**39.79**++
SYS2	*Baseline*	38.84	38.30
	Baseline+mLM	38.95*	38.63+
	FS-Comb	**39.92**++	**39.49**++

Table 6: Translation results of *Baseline*, *Baseline+mLM* and *FS-Comb* (+: significant better than baseline system with $p < 0.05$; ++: significant better than baseline system with $p < 0.01$; *: no significant improvement).

We also investigate the results when we incrementally add the *n*-best list of each sub-system into a candidate pool to see the effects when different numbers of sub-systems are used in combination. In order to decide the sequence of sub-systems to add, we first evaluate the performance of pair-wise combinations between each sub-system and its baseline system on the development set. That is, for each sub-system, we combine its *n*-best list with the *n*-best list of its baseline system and perform system combination for MT03 data set. Then we rank the sub-systems by the pair-wise combination performance from high to low, and use this ranking as the sequence to add *n*-best lists of sub-systems. Each time when a new *n*-best list is added, the combination performance based on the enlarged candidate pool is evaluated. Figure 2 shows the results on both MT04 and MT05 test sets, in which ***SYS1-fs*** and ***SYS2-fs*** denote the sub-systems derived from SYS1 and SYS2 respectively, and X-axis is the number of sub-systems used for combination each time and Y-axis is the BLEU score. From the figure we can see that although in some cases the performance slightly drops when a new sub-system is added, generally using more sub-systems always leads to better results.

Next we examine the performance of baseline systems when different beam sizes are used in decoding. The results on MT05 test set are shown in Figure3, where X-axis is the beam size. In Figure 3, ***SYS1+mLM*** and ***SYS2+mLM*** denote augmented baseline systems of SYS1 and SYS2 with multiple LM features.

From Figure 3 we can see that augmented baseline systems (with multiple LM features) outperform the baseline systems (with only one LM feature) for all beam sizes ranging from 20 to 220. In this experiment we did not observe any significant performance improvements when using larger beam sizes than the default setting, but using more sub-systems in combination almost always bring improvements.

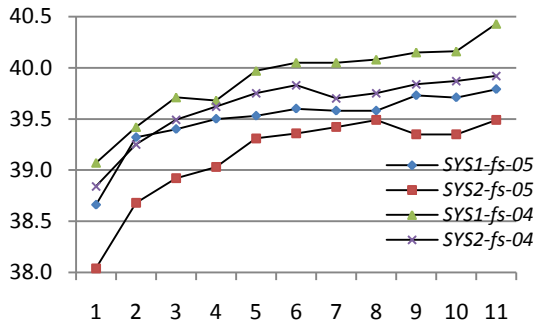


Figure 2: Performances on different numbers of sub-systems.

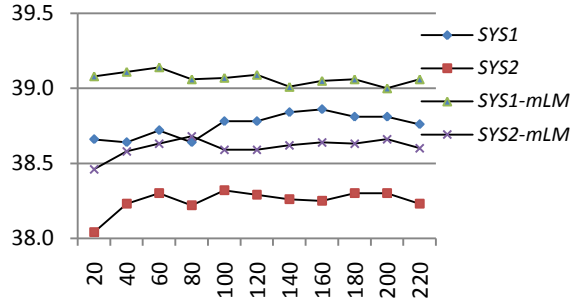


Figure 3: Performances on different beam sizes.

	MT04	MT05
SYS1-fs	44.63%	46.12%
SYS2-fs	47.54%	44.73%

Table 7: Ratio of final translations coming from non-baseline sub-systems.

Finally, we investigate the ratio of final translations coming from the *n*-best lists of non-baseline sub-systems only. Table 7 shows the results on both MT04 and MT05 test sets, which indicate that almost half of the final translations are contributed by the non-baseline sub-systems.

4.5 The Impact of *n*-best List Size

In order to find the optimal size of *n*-best list for combination, we compare the combination results of using list sizes from 10-best up to 500-best for each sub-system.

In this experiment, system combination was performed on the combined *n*-best list from total 11 sub-systems with different list size each time. Figure 4 shows the results on the MT03 dev set and the MT04 and MT05 test sets for both SYS1 and SYS2. X-axis is the *n*-best list size of each sub-system.

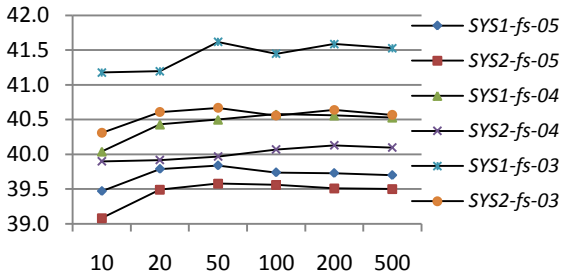


Figure 4: Performances on different *n*-best sizes.

We can see from the figure that for all data sets the optimal *n*-best list size is around 50, but the improvements are not significant over the results when 20-best translations are used. The reason for the small optimal *n*-best list size could be that the low-rank hypotheses might introduce more noises into the combined translation candidate pool for sentence-level combination (Hasan et al., 2007; Hildebrand and Vogel, 2008).

4.6 Feature Subspace Method on Multiple SMT Systems

In the last experiment, we investigate the effect of feature subspace method when multiple SMT systems are used in system combination.

Evaluation results are reported in Table 8. The system combination method described in Section 3.2 is used to combine outputs from two baseline systems (with only one 5-gram LM feature) and sub-systems generated from both baseline systems (22 in total), with their results denoted as ***Baseline Comb* (*both*)** and ***FS Comb* (*both*)** respectively. We also include the combination results of sub-systems based on one baseline system for reference in the table.

On both MT04 and MT05 test sets, the results of system combination based on sub-systems are significantly better than those of baseline systems, which show that our method can also help with system combination when more than one system are used. We can also see that using multiple systems based on different SMT models and using our subspace based method can help each other: the best performance can only be achieved when both are employed.

	MT04	MT05
Baseline Comb (*both*)	39.98	39.43
FS-Comb (SYS1)	40.43	39.79
FS-Comb (SYS2)	39.92	39.49
FS Comb (*both*)	**40.96**	**40.38**

Table 8: Performances of sentence-level combination on multiple SMT systems.

5 Conclusion

In this paper, we have presented a novel and effective *Feature Subspace* method for the construction of an ensemble of machine translation systems based on a baseline SMT model which can be formulated as a standard linear function. Each system within the ensemble is based on a subset of features derived from the baseline model, and the resulting ensemble can be used in system combination to improve translation quality. Experimental results on NIST Chinese-to-English translation tasks show that our method can bring significant improvements to two baseline systems with state-of-the-art performance, and it is expected that our method can be employed to improve any linear model -based SMT systems. There is still much room for improvements in the current work. For example, we still use a simple one-feature difference principle for feature subspace generation. In the future, we will explore more possibilities for feature subspaces selection and experiment with our method in a word-level system combination model.

References

Necip Fazil Ayan, Jing Zheng, and Wen Wang. 2008. Improving alignments for better confusion networks for combining machine translation systems. In *Proc. COLING*, pages 33-40.

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Proc. ASRU*, pages 351-354.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL*, pages 263-270.

Xiaodong He, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore. 2008. Indirect-hmm-based hypothesis for combining outputs from machine translation systems. In *Proc. EMNLP*, pages 98-107.

Almut Silja Hildebrand and Stephan Vogel. 2008. Combination of machine translation systems via hypothesis selection from combined n-best lists. In *8th AMTA conference*, pages 254-261.

Tin Kam Ho. 1998. The random subspace method for constructing decision forests. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 832-844.

Sasa Hasan, Richard Zens, and Hermann Ney. 2007. Are very large *n*-best lists useful for SMT? In *Proc. NAACL, Short paper*, pages 57-60.

S. Jayaraman and A. Lavie. 2005. Multi-Engine Machine Translation Guided by Explicit Word Matching. In *10th EAMT conference*, pages 143-152.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP*, pages 388-395.

Philipp Koehn. 2004. Phrase-based Model for SMT. In *Computational Linguistics*, 28(1): pages 114-133.

Mu Li, Nan Duan, Dongdong Zhang, Chi-Ho Li, and Ming Zhou. 2009. Collaborative Decoding: Partial Hypothesis Re-Ranking Using Translation Consensus between Decoders. In *Proc. ACL-IJCNLP*.

Adam Lopez and Philip Resnik. 2006. Word-Based Alignment, Phrase-Based Translation: What's the link? In *7th AMTA conference*, pages 90-99.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-String Alignment Template for Statistical Machine Translation. In *Proc. ACL*, pages 609-616.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. EMNLP*, pages 44-52.

Wolfgang Macherey and Franz Och. 2007. An Empirical Study on Computing Consensus Translations from Multiple Machine Translation Systems. In *Proc. EMNLP*, pages 986-995.

Evgeny Matusov, Nicola Ueffi ng, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proc. EACL*, pages 33-40.

Franz Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statis-

tical machine translation. In *Proc. ACL*, pages 295-302.

Franz Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL*, pages 160-167.

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

Franz Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4): pages 417-449.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. 2007. Combining outputs from multiple machine translation systems. In *Proc. NAACL*, pages 228-235.

Antti-Veikko Rosti, Spyros Matsoukas, and Richard Schwartz. 2007. Improved Word-Level System Combination for Machine Translation. In *Proc. ACL*, pages 312-319.

Antti-Veikko Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2008. Incremental hypothesis alignment for building confusion networks with application to machine translation system combination. In *Proc. Of the Third ACL Workshop on Statistical Machine Translation*, pages 183-186.

K.C. Sim, W. Byrne, M. Gales, H. Sahbi, and P. Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *ICASSP*, pages 105-108.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proc. ACL*, pages 521-528.

Yang Ye, Ming Zhou, and Chin-Yew Lin. 2007. Sentence level Machine Translation Evaluation as a Ranking Problem: One step aside from BLEU. In *Proc. Of the Second ACL Workshop on Statistical Machine Translation*, pages 240-247.

Lattice-based System Combination for Statistical Machine Translation

Yang Feng, Yang Liu, Haitao Mi, Qun Liu, Yajuan Lü
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
{fengyang, yliu, htmi, liuqun, lvyajuan}@ict.ac.cn

Abstract

Current system combination methods usually use confusion networks to find consensus translations among different systems. Requiring one-to-one mappings between the words in candidate translations, confusion networks have difficulty in handling more general situations in which several words are connected to another several words. Instead, we propose a lattice-based system combination model that allows for such phrase alignments and uses *lattice*s to encode all candidate translations. Experiments show that our approach achieves significant improvements over the state-of-the-art baseline system on Chinese-to-English translation test sets.

1 Introduction

System combination aims to find consensus translations among different machine translation systems. It has been proven that such consensus translations are usually better than the output of individual systems (Frederking and Nirenburg, 1994).

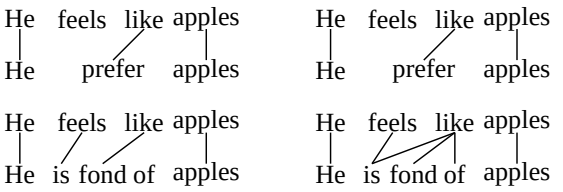
In recent several years, the system combination methods based on confusion networks developed rapidly (Bangalore et al., 2001; Matusov et al., 2006; Sim et al., 2007; Rosti et al., 2007a; Rosti et al., 2007b; Rosti et al., 2008; He et al., 2008), which show state-of-the-art performance in benchmarks. A confusion network consists of a sequence of sets of candidate words. Each candidate word is associated with a score. The optimal consensus translation can be obtained by selecting one word from each set to maximizing the overall score.

To construct a confusion network, one first need to choose one of the hypotheses (i.e., candidate translations) as the backbone (also called "skeleton" in the literature) and then decide the word alignments of other hypotheses to the backbone. Hypothesis alignment plays a crucial role in confusion-network-based system combination because it has a direct effect on selecting consensus translations.

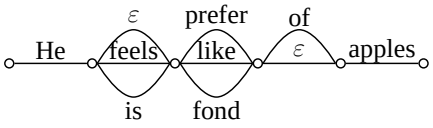
However, a confusion network is restricted in such a way that only 1-to-1 mappings are allowed in hypothesis alignment. This is not the fact even for word alignments between the same languages. It is more common that several words are connected to another several words. For example, "be capable of" and "be able to" have the same meaning. Although confusion-network-based approaches resort to inserting null words to alleviate this problem, they face the risk of producing degenerate translations such as "be capable to" and "be able of".

In this paper, we propose a new system combination method based on lattices. As a more general form of confusion network, a lattice is capable of describing arbitrary mappings in hypothesis alignment. In a lattice, each edge is associated with a sequence of words rather than a single word. Therefore, we select phrases instead of words in each candidate set and minimize the chance to produce unexpected translations such as "be capable to". We compared our approach with the state-of-the-art confusion-network-based system (He et al., 2008) and achieved a significant absolute improvement of 1.23 BLEU points on the NIST 2005 Chinese-to-English test set and 0.93 BLEU point on the NIST 2008 Chinese-to-English test set.

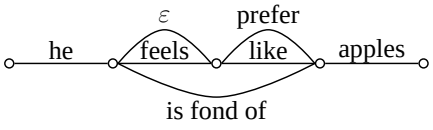
Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1105–1113,
Singapore, 6-7 August 2009.



(a) unidirectional alignments (b) bidirectional alignments



(c) confusion network



(d) lattice

Figure 1: Comparison of a confusion network and a lattice.

2 Background

2.1 Confusion Network and Lattice

We use an example shown in Figure 1 to illustrate our idea. Suppose that there are three hypotheses:

He feels like apples
He prefer apples
He is fond of apples

We choose the first sentence as the backbone. Then, we perform hypothesis alignment to build a confusion network, as shown in Figure 1(a). Note that although "*feels like*" has the same meaning with "*is fond of*", a confusion network only allows for one-to-one mappings. In the confusion network shown in Figure 1(c), several null words ε are inserted to ensure that each hypothesis has the same length. As each edge in the confusion network only has a single word, it is possible to produce inappropriate translations such as "*He is like of apples*".

In contrast, we allow many-to-many mappings in the hypothesis alignment shown in Figure 2(b). For example, "*like*" is aligned to three words: "*is*", "*fond*", and "*of*". Then, we use a lattice shown in Figure 1(d) to represent all possible candidate translations. Note that the phrase "*is fond of*" is attached to an edge. Now, it is unlikely to obtain a translation like "*He is like of apples*".

A lattice $G = \langle V, E \rangle$ is a directed acyclic graph, formally a weighted finite state automation (FSA), where V is the set of nodes and E is the set of edges. The nodes in a lattice are usually labeled according to an appropriate numbering to reflect how to produce a translation. Each edge in a lattice is attached with a sequence of words as well as the associated probability.

As lattice is a more general form of confusion network (Dyer et al., 2008), we expect that replacing confusion networks with lattices will further improve system combination.

2.2 IHMM-based Alignment Method

Since the candidate hypotheses are aligned using Indirect-HMM-based (IHMM-based) alignment method (He et al., 2008) in both direction, we briefly review the IHMM-based alignment method first. Take the direction that the hypothesis is aligned to the backbone as an example. The conditional probability that the hypothesis is generated by the backbone is given by

$$p({e'_1}^J|e_1^I) = \sum_{a_1^J} \prod_{j=1}^{J} [p(a_j|a_{j-1}, I) p(e'_j|e_{a_j})] l \quad (1)$$

Where $e_1^I = (e_1, ..., e_I)$ is the backbone, ${e'}_1^J = (e'_1, ..., e'_J)$ is a hypothesis aligned to e_1^I, and $a_1^J = (a_1, .., a_J)$ is the alignment that specifies the position of backbone word that each hypothesis word is aligned to.

The translation probability $p(e'_j|e_i)$ is a linear interpolation of semantic similarity $p_{sem}(e'_j|e_i)$ and surface similarity $p_{sur}(e'_j|e_i)$ and α is the interpolation factor:

$$p(e'_j|e_i) = \alpha \cdot p_{sem}(e'_j|e_i) + (1-\alpha) \cdot p_{sur}(e'_j|e_i) \quad (2)$$

The semantic similarity model is derived by using the source word sequence as a hidden layer, so the bilingual dictionary is necessary. The semantic sim-

ilarity model is given by

$$p_{sem}(e'_j|e_i) = \sum_{k=0}^{K} p(f_k|e_i)p(e'_j|f_k, e_i) \approx \sum_{k=0}^{K} p(f_k|e_i)p(e'_j|f_k) \quad (3)$$

The surface similarity model is estimated by calculating the literal matching rate:

$$p_{sur}(e'_j|e_i) = \exp\{\rho \cdot [s(e'_j, e_i) - 1]\} \quad (4)$$

where $s(e'_j, e_i)$ is given by

$$s(e'_j, e_i) = \frac{M(e'_j, e_i)}{\max(|e'_j|, |e_i|)} \quad (5)$$

where $M(e'_j, e_i)$ is the length of the longest matched prefix (LMP) and ρ is a smoothing factor that specifies the mapping.

The distortion probability $p(a_j = i|a_{j-1} = i', I)$ is estimated by only considering the jump distance:

$$p(i|i', I) = \frac{c(i - i')}{\sum_{i=1}^{I} c(l - i')} \quad (6)$$

The distortion parameters $c(d)$ are grouped into 11 buckets, $c(\leq -4)$, $c(-3)$, ..., $c(0)$, ..., $c(5)$, $c(\geq 6)$. Since the alignments are in the same language, the distortion model favor monotonic alignments and penalize non-monotonic alignments. It is given in a intuitive way

$$c(d) = (1 + |d - 1|)^{-K}, d = -4, ..., 6 \quad (7)$$

where K is tuned on held-out data.

Also the probability p_0 of jumping to a $null$ word state is tuned on held-out data. So the overall distortion model becomes

$$p(i|i', I) = \begin{cases} p_0 & \text{if i = null state} \\ (1 - p_0) \cdot p(i|i', I) & \text{otherwise} \end{cases}$$

3 Lattice-based System Combination Model

Lattice-based system combination involves the following steps:

(1) Collect the hypotheses from the candidate systems.

(2) Choose the backbone from the hypotheses. This is performed using a sentence-level Minimum Bayes Risk (MBR) method. The hypothesis with the minimum cost of edits against all hypotheses is selected. The backbone is significant for it influences not only the word order, but also the following alignments. The backbone is selected as follows:

$$E_B = \underset{E' \in \mathbf{E}}{\operatorname{argmin}} \sum_{E \in \mathbf{E}} TER(E', E) \quad (8)$$

(3) Get the alignments of the backbone and hypothesis pairs. First, each pair is aligned in both directions using the IHMM-based alignment method. In the IHMM alignment model, bilingual dictionaries in both directions are indispensable. Then, we apply a grow-diag-final algorithm which is widely used in bilingual phrase extraction (Koehn et al., 2003) to monolingual alignments. The bidirectional alignments are combined to one resorting to the grow-diag-final algorithm, allowing n-to-n mappings.

(4)Normalize the alignment pairs. The word order of the backbone determines the word order of consensus outputs, so the word order of hypotheses must be consistent with that of the backbone. All words of a hypotheses are reordered according to the alignment to the backbone. For a word aligned to $null$, an actual $null$ word may be inserted to the proper position. The *alignment unit*s are extracted first and then the hypothesis words in each unit are shifted as a whole.

(5) Construct the lattice in the light of phrase pairs extracted on the normalized alignment pairs. The expression ability of the lattice depends on the phrase pairs.

(6) Decode the lattice using a model similar to the log-linear model.

The confusion-network-based system combination model goes in a similar way. The first two steps are the same as the lattice-based model. The difference is that the hypothesis pairs are aligned just in one direction due to the expression limit of the confusion network. As a result, the normalized alignments only contain 1-to-1 mappings (Actual $null$ words are also needed in the case of null alignment). In the following, we will give more details about the steps which are different in the two models.

4 Lattice Construction

Unlike a confusion network that operates words only, a lattice allows for phrase pairs. So phrase pairs must be extracted before constructing a lattice. A major difficulty in extracting phrase pairs is that the word order of hypotheses is not consistent with that of the backbone. As a result, hypothesis words belonging to a phrase pair may be discontinuous. Before phrase pairs are extracted, the hypothesis words should be normalized to make sure the words in a phrase pair is continuous. We call a phrase pair before normalization a *alignment unit*.

The problem mentioned above is shown in Figure 2. In Figure 2 (a), although ($e'_1e'_3$, e_2) should be a phrase pair, but "e'_1" and "e'_3" are discontinuous, so the phrase pair can not be extracted. Only after the words of the hypothesis are reordered according to the corresponding words in the backbone as shown in Figure 2 (b), "e'_1" and "e'_3" become continuous and the phrase pair ($e'_1e'_3$, e_2) can be extracted. The procedure of reordering is called *alignment normalization*

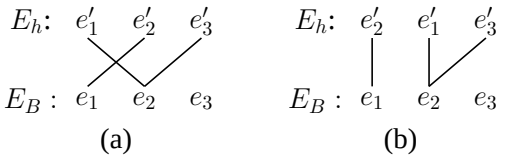


Figure 2: An example of alignment units

4.1 Alignment Normalization

After the final alignments are generated in the grow-diag-final algorithm, *minimum alignment unit*s are extracted. The hypothesis words of an alignment unit are packed as a whole in shift operations.

See the example in Figure 2 (a) first. All minimum alignment units are as follows: (e'_2, e_1), ($e'_1e'_3$, e_2) and (ε, e_3). ($e'_1e'_2e'_3$, e_1e_2) is an alignment unit, but not a minimum alignment unit.

Let $\bar{a}_i = (\bar{e}'_i, \bar{e}_i)$ denote a minimum alignment unit, and assume that the word string $\bar{e}'_i$ covers words e'_{i_1} ,..., e'_{i_m} on the hypothesis side, and the word string $\bar{e}_i$ covers the consecutive words e_{i_1} ,..., e_{i_n} on the backbone side. In an alignment unit, the word string on the hypothesis side can be discontinuous. The minimum unit $\bar{a}_i = (\bar{e}'_i, \bar{e}_i)$ must observe the following rules:

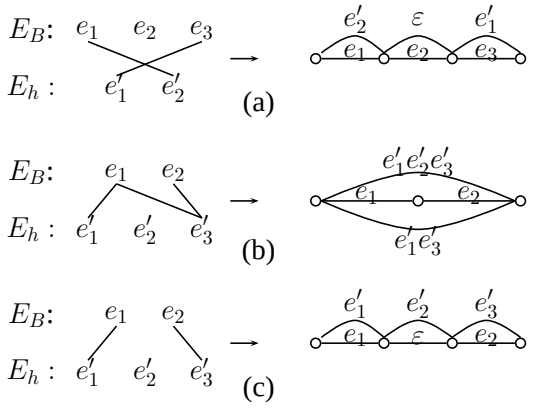


Figure 3: Different cases of $null$ insertion

- $\forall\ e'_{i_k} \in \bar{e}'_i, \quad e_{a'_{i_k}} \in \bar{e}_i$
- $\forall\ e_{i_k} \in \bar{e}_i, \quad e'_{a_{i_k}} = null$ or $e'_{a_{i_k}} \in \bar{e}'_i$
- $\nexists\ \bar{a}_j = (\bar{e}'_j, \bar{e}_j), \quad \bar{e}_j = e_{i_1}, ..., e_{i_k}$ or $\bar{e}_j = e_{i_k}, ..., e_{i_n}, \quad k \in [1, n]$

Where a'_{i_k} denotes the position of the word in the backbone that e'_{i_k} is aligned to, and a_{i_k} denotes the position of the word in the hypothesis that e_{i_k} is aligned to.

An actual $null$ word may be inserted to a proper position if a word, either from the hypothesis or from the backbone, is aligned to $null$. In this way, the minimum alignment set is extended to an alignment unit set, which includes not only minimum alignment units but also alignment units which are generated by adding $null$ words to minimum alignment units. In general, the following three conditions should be taken into consideration:

- A backbone word is aligned to $null$. A $null$ word is inserted to the hypothesis as shown in Figure 3 (a).
- A hypothesis word is aligned to $null$ and it is between the span of a minimum alignment unit. A new alignment unit is generated by inserting the hypothesis word aligned to null to the minimum alignment unit. The new hypothesis string must remain the original word order of the hypothesis. It is illustrated in Figure 3 (b).
- A hypothesis word is aligned to $null$ and it is not between the hypothesis span of any minimum alignment unit. In this case, a $null$ word

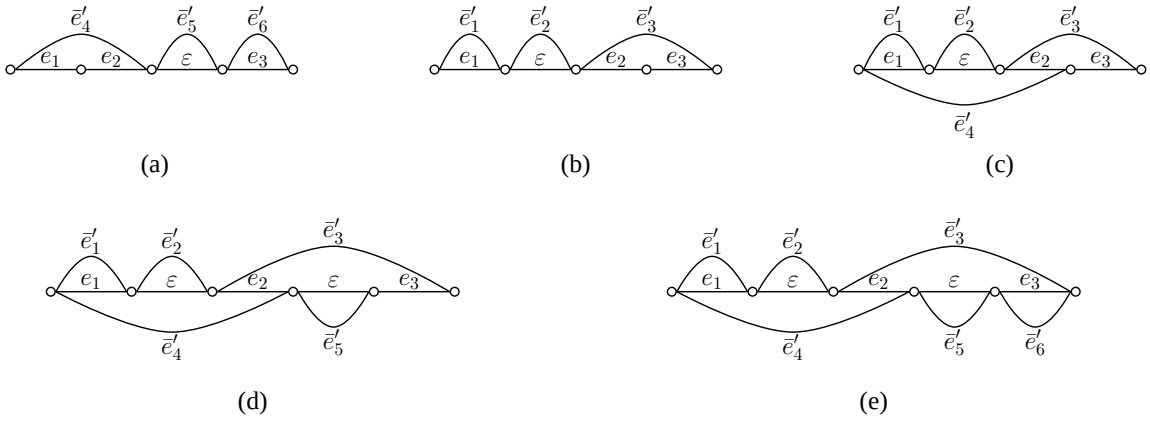


Figure 4: A toy instance of lattice construction

are inserted to the backbone. This is shown in Figure 3 (c).

4.2 Lattice Construction Algorithm

The lattice is constructed by adding the normalized alignment pairs incrementally. One backbone arc in a lattice can only span one backbone word. In contrast, all hypothesis words in an alignment unit must be packed into one hypothesis arc. First the lattice is initialized with a normalized alignment pair. Then given all other alignment pairs one by one, the lattice is modified dynamically by adding the hypothesis words of an alignment pair in a left-to-right fashion.

A toy instance is given in Figure 4 to illustrate the procedure of lattice construction. Assume the current inputs are: an alignment pair as in Figure 4 (a), and a lattice as in Figure 4 (b). The backbone words of the alignment pair are compared to the backbone words of the lattice one by one. The procedure is as follows:

- e_1 is compared with e_1. Since they are the same, the hypothesis arc $\bar{e}'_4$, which comes from the same node with e_1 in the alignment pair, is compared with the hypothesis arc $\bar{e}'_1$, which comes from the same node with e_1 in the lattice. The two hypothesis arcs are not the same, so $\bar{e}'_4$ is added to the lattice as shown in Figure 4(c). Both go to the next backbone words.

- e_2 is compared with ε. The lattice remains the same. The lattice goes to the next backbone word e_2.

- e_2 is compared with e_2. There is no hypothesis arc coming from the same node with the bone arc e_2 in the alignment pair, so the lattice remains the same. Both go to the next backbone words.

- ε is compared with e_3. A $null$ backbone arc is inserted into the lattice between e_2 and e_3. The hypothesis arc $\bar{e}'_5$ is inserted to the lattice, too. The modified lattice is shown in Figure 4(d). The alignment pair goes to the next backbone word e_3.

- e_3 is compared with e_3. For they are the same and there is no hypothesis arc $\bar{e}'_6$ in the lattice, $\bar{e}'_6$ is inserted to the lattice as in Figure 4(e).

- Both arrive at the end and it is the turn of the next alignment pair.

When comparing a backbone word of the given alignment pair with a backbone word of the lattice, the following three cases should be handled:

- The current backbone word of the given alignment pair is a $null$ word while the current backbone word of the lattice is not. A $null$ backbone word is inserted to the lattice.

- The current backbone word of the lattice is a $null$ word while the current word of the given alignment pair is not. The current $null$ backbone word of the lattice is skipped with nothing to do. The next backbone word of the lattice is compared with the current backbone word of the given alignment pair.

Algorithm 1 Lattice construction algorithm.

1: **Input**: alignment pairs $\{p_n\}_{n=1}^{N}$
2: $L \leftarrow p_1$
3: $Unique(L)$
4: **for** $n \leftarrow 2$ .. N **do**
5: $pnode = p_n \cdot first$
6: $lnode = L \cdot first$
7: **while** $pnode \cdot barcnext \neq NULL$ **do**
8: **if** $lnode \cdot barcnext = NULL$ or $pnode \cdot bword = null$ and $lnode \cdot bword \neq null$ **then**
9: INSERTBARC($lnode$, $null$)
10: $pnode = pnode \cdot barcnext$
11: **else**
12: **if** $pnode \cdot bword \neq null$ and $lnode \cdot bword = null$ **then**
13: $lnode = lnode \cdot barcnext$
14: **else**
15: **for** each $harc$ of $pnode$ **do**
16: **if** NotExist($lnode, pnode \cdot harc$) **then**
17: INSERTHARC($lnode$, $pnode \cdot harc$)
18: $pnode = pnode \cdot barcnext$
19: $lnode = lnode \cdot barcnext$
20: **Output:** lattice L

- The current backbone words of the given alignment pair and the lattice are the same. Let $\{harc_l\}$ denotes the set of hypothesis arcs, which come from the same node with the current backbone arc in the lattice, and $harc_h$ denotes one of the corresponding hypothesis arcs in the given alignment pair. In the $\{harc_l\}$, if there is no arc which is the same with the $harc_h$, a hypothesis arc projecting to $harc_h$ is added to the lattice.

The algorithm of constructing a lattice is illustrated in Algorithm 1. The backbone words of the alignment pair and the lattice are processed one by one in a left-to-right manner. Line 2 initializes the lattice with the first alignment pair, and Line 3 removes the hypothesis arc which contains the same words with the backbone arc. $barc$ denotes the backbone arc, storing one backbone word only, and $harc$ denotes the hypothesis arc, storing the hypothesis words. For there may be many alignment units span the same backbone word range, there may be more than one $harc$ coming from one node. Line $8-10$ consider the condition 1 and function *InsertBarc* in Line 9 inserts a $null$ bone arc to the position right before the current node. Line $12-13$ deal with condition 2 and jump to the next backbone word of the lattice. Line $15-19$ handle condition 3 and function *InsertHarc* inserts to the lattice a $harc$ with the same hypothesis words and the same backbone word span with the current hypothesis arc.

5 Decoding

In confusion network decoding, a translation is generated by traveling all the nodes from left to right. So a translation path contains all the nodes. While in lattice decoding, a translation path may skip some nodes as some hypothesis arcs may cross more than one backbone arc.

Similar to the features in Rosti et al. (2007a), the features adopted by lattice-based model are arc posterior probability, language model probability, the number of $null$ arcs, the number of hypothesis arcs possessing more than one non-null word and the number of all non-null words. The features are combined in a log-linear model with the arc posterior probabilities being processed specially as follows:

$$\begin{aligned}\log p(\mathbf{e}/\mathbf{f}) = &\sum_{i=1}^{N_{arc}} \log (\sum_{s=1}^{N_s} \lambda_s p_s(arc)) \\ &+ \zeta L(\mathbf{e}) + \alpha N_{nullarc}(\mathbf{e}) \\ &+ \beta N_{longarc}(\mathbf{e}) + \gamma N_{word}(\mathbf{e})\end{aligned} \tag{9}$$

where $\mathbf{f}$ denotes the source sentence, $\mathbf{e}$ denotes a translation generated by the lattice-based system, N_{arc} is the number of arcs the path of $\mathbf{e}$ covers, N_s is the number of candidate systems and λ_s is the weight of system s. ζ is the language model weight and $L(\mathbf{e})$ is the LM log-probability. $N_{nullarcs}(\mathbf{e})$ is the number of the arcs which only contain a $null$ word, and $N_{longarc}(\mathbf{e})$ is the number of the arcs which store more than one non-null word. The above two numbers are gotten by counting both backbone arcs and hypothesis arcs. α and β are the corresponding weights of the numbers, respectively. $N_{word}(\mathbf{e})$ is the non-null word number and γ is its weight.

Each arc has different confidences concerned with different systems, and the confidence of system s is denoted by $p_s(arc)$. $p_s(arc)$ is increased by

$1/(k+1)$ if the hypothesis ranking k in the system s contains the arc (Rosti et al., 2007a; He et al., 2008).

Cube pruning algorithm with beam search is employed to search for the consensus output (Huang and Chiang, 2005). The nodes in the lattice are searched in a topological order and each node retains a list of N best candidate partial translations.

6 Experiments

The candidate systems participating in the system combination are as listed in Table 1: System A is a BTG-based system using a MaxEnt-based reordering model; System B is a hierarchical phrase-based system; System C is the Moses decoder (Koehn et al., 2007); System D is a syntax-based system. 10-best hypotheses from each candidate system on the dev and test sets were collected as the input of the system combination.

In our experiments, the weights were all tuned on the NIST MT02 Chinese-to-English test set, including 878 sentences, and the test data was the NIST MT05 Chinese-to-English test set, including 1082 sentences, except the experiments in Table 2. A 5-gram language model was used which was trained on the XinHua portion of Gigaword corpus. The results were all reported in case sensitive BLEU score and the weights were tuned in Powell's method to maximum BLEU score. The IHMM-based alignment module was implemented according to He et al. (2008), He (2007) and Vogel et al. (1996). In all experiments, the parameters for IHMM-based alignment module were set to: the smoothing factor for the surface similarity model, $\rho = 3$; the controlling factor for the distortion model, $K = 2$.

6.1 Comparison with Confusion-network-based model

In order to compare the lattice-based system with the confusion-network-based system fairly, we used IHMM-based system combination model on behalf of the confusion-network-based model described in He et al. (2008). In both lattice-based and IHMM-based systems, the bilingual dictionaries were extracted on the FBIS data set which included 289K sentence pairs. The interpolation factor of the similarity model was set to $\alpha = 0.1$.

The results are shown in Table 1. *IHMM* stands for the IHMM-based model and *Lattice* stands for the lattice-based model. On the dev set, the lattice-based system was 3.92 BLEU points higher than the best single system and 0.36 BLEU point higher than the IHMM-based system. On the test set, the lattice-based system got an absolute improvement by 3.73 BLEU points over the best single system and 1.23 BLEU points over the IHMM-based system.

System	MT02 BLEU%	MT05 BLEU%
SystemA	31.93	30.68
SystemB	32.16	32.07
SystemC	32.09	31.64
SystemD	33.37	31.26
IHMM	36.93	34.57
Lattice	37.29	35.80

Table 1: Results on the MT02 and MT05 test sets

The results on another test sets are reported in Table 2. The parameters were tuned on the newswire part of NIST MT06 Chinese-to-English test set, including 616 sentences, and the test set was NIST MT08 Chinese-to-English test set, including 1357 sentences. The BLEU score of the lattice-based system is 0.93 BLEU point higher than the IHMM-based system and 3.0 BLEU points higher than the best single system.

System	MT06 BLEU%	MT08 BLEU%
SystemA	32.51	25.63
SystemB	31.43	26.32
SystemC	31.50	23.43
SystemD	32.41	26.28
IHMM	36.05	28.39
Lattice	36.53	29.32

Table 2: Results on the MT06 and MT08 test sets

We take a real example from the output of the two systems (in Table 3) to show that higher BLEU scores correspond to better alignments and better translations. The translation of System C is selected as the backbone. From Table 3, we can see that because of 1-to-1 mappings, "*Russia*" is aligned to "*Russian*" and "*'s*" to "*null*" in the IHMM-based model, which leads to the error translation "*Russian*

Source:	俄罗斯 国营 石油 公司 正 与 俄罗斯 国营 瓦斯 公司 进行合并
SystemA:	Russia merger of state-owned oil company and the state-run gas company in Russia
SystemB:	Russia 's state-owned oil company is working with Russia 's state-run gas company mergers
SystemC:	Russian state-run oil company is combined with the Russian state-run gas company
SystemD:	Russia 's state-owned oil companies are combined with Russia 's state-run gas company
IHMM:	Russian 's state-owned oil company working with Russia 's state-run gas company
Lattice:	Russia 's state-owned oil company is combined with the Russian state-run gas company

Table 3: A real translation example

's". Instead, "*Russia 's*" is together aligned to "*Russian*" in the lattice-based model. Also due to 1-to-1 mappings, $null$ word aligned to "*is*" is inserted. As a result, "*is*" is missed in the output of IHMM-based model. In contrast, in the lattice-based system, "*is working with*" are aligned to "*is combined with*", forming a phrase pair.

6.2 Effect of Dictionary Scale

The dictionary is important to the semantic similarity model in IHMM-based alignment method. We evaluated the effect of the dictionary scale by using dictionaries extracted on different data sets. The dictionaries were respectively extracted on similar data sets: 30K sentence pairs, 60K sentence pairs, 289K sentence pairs (FBIS corpus) and 2500K sentence pairs. The results are illustrated in Table 4. In order to demonstrate the effect of the dictionary size clearly, the interpolation factor of similarity model was all set to $\alpha = 0.1$.

From Table 4, we can see that when the corpus size rise from 30k to 60k, the improvements were not obvious both on the dev set and on the test set. As the corpus was expanded to 289K, although on the dev set, the result was only 0.2 BLEU point higher, on the test set, it was 0.63 BLEU point higher. As the corpus size was up to 2500K, the BLEU scores both on the dev and test sets declined. The reason is that, on one hand, there are more noise on the 2500K sentence pairs; on the other hand, the 289K sentence pairs cover most of the words appearing on the test set. So we can conclude that in order to get better results, the dictionary scale must be up to some certain scale. If the dictionary is much smaller, the result will be impacted dramatically.

	MT02 BLEU%	MT05 BLEU%
30k	36.94	35.14
60k	37.09	35.17
289k	37.29	35.80
2500k	37.14	35.62

Table 4: Effect of dictionary scale

6.3 Effect of Semantic Alignments

For the IHMM-based alignment method, the translation probability of an English word pair is computed using a linear interpolation of the semantic similarity and the surface similarity. So the two similarity models decide the translation probability together and the proportion is controlled by the interpolation factor. We evaluated the effect of the two similarity models by varying the interpolation factor α.

We used the dictionaries extracted on the FBIS data set. The result is shown in Table 5. We got the best result with $\alpha = 0.1$. When we excluded the semantic similarity model ($\alpha = 0.0$) or excluded the surface similarity model ($\alpha = 1.0$), the performance became worse.

7 Conclusion

The alignment model plays an important role in system combination. Because of the expression limitation of confusion networks, only 1-to-1 mappings are employed in the confusion-network-based model. This paper proposes a lattice-based system combination model. As a general form of confusion networks, lattices can express n-to-n mappings. So a lattice-based model processes phrase pairs while

	MT02 BLEU%	MT05 BLEU%
$\alpha = 1.0$	36.41	34.92
$\alpha = 0.7$	37.21	35.65
$\alpha = 0.5$	36.43	35.02
$\alpha = 0.4$	37.14	35.55
$\alpha = 0.3$	36.75	35.66
$\alpha = 0.2$	36.81	35.55
$\alpha = 0.1$	37.29	35.80
$\alpha = 0.0$	36.45	35.14

Table 5: Effect of semantic alignments

a confusion-network-based model processes words only. As a result, phrase pairs must be extracted before constructing a lattice.

On NIST MT05 test set, the lattice-based system gave better results with an absolute improvement of 1.23 BLEU points over the confusion-network-based system (He et al., 2008) and 3.73 BLEU points over the best single system. On NIST MT08 test set, the lattice-based system outperformed the confusion-network-based system by 0.93 BLEU point and outperformed the best single system by 3.0 BLEU points.

8 Acknowledgement

The authors were supported by National Natural Science Foundation of China Contract 60736014, National Natural Science Foundation of China Contract 60873167 and High Technology R&D Program Project No. 2006AA010108. Thank Wenbin Jiang, Tian Xia and Shu Cai for their help. We are also grateful to the anonymous reviewers for their valuable comments.

References

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Proc. of IEEE ASRU*, pages 351–354.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL/HLT 2008*, pages 1012–1020, Columbus, Ohio, June.

Robert Frederking and Sergei Nirenburg. 1994. Three heads are better than one. In *Proc. of ANLP*, pages 95–100.

Xiaodong He, Mei Yang, Jangfeng Gao, Patrick Nguyen, and Robert Moore. 2008. Indirect-hmm-based hypothesis alignment for computing outputs from machine translation systems. In *Proc. of EMNLP*, pages 98–107.

Xiaodong He. 2007. Using word-dependent translation models in hmm based word alignment for statistical machine translation. In *Proc. of COLING-ACL*, pages 961–968.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT)*, pages 53–64.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL*, pages 127–133.

Philipp Koehn, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of the 45th ACL, Demonstration Session*.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proc. of IEEE EACL*, pages 33–40.

Antti-Veikko I. Rosti, Spyros Matsoukas, and Richard Schwartz. 2007a. Improved word-level system combination for machine translation. In *Proc. of ACL*, pages 312–319.

Antti-Veikko I. Rosti, Bing Xiang, Spyros Matsoukas, Richard Schwartz, Necip Fazil Ayan, and Bonnie J. Dorr. 2007b. Combining outputs from multiple machine translation systems. In *Proc. of NAACL-HLT*, pages 228–235.

Antti-Veikko I. Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2008. Incremental hypothesis alignment for building confusion networks with application to machine translaiton system combination. In *Proc. of the Third ACL WorkShop on Statistical Machine Translation*, pages 183–186.

Khe Chai Sim, William J. Byrne, Mark J.F. Gales, Hichem Sahbi, and Phil C. Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *Proc. of ICASSP*, pages 105–108.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proc. of COLING*, pages 836–841.

A Joint Language Model With Fine-grain Syntactic Tags

Denis Filimonov[1]
[1]Laboratory for Computational Linguistics and Information Processing
Institute for Advanced Computer Studies
University of Maryland, College Park
den@cs.umd.edu

Mary Harper[1,2]
[2]Human Language Technology Center of Excellence
Johns Hopkins University
mharper@umiacs.umd.edu

Abstract

We present a scalable joint language model designed to utilize fine-grain syntactic tags. We discuss challenges such a design faces and describe our solutions that scale well to large tagsets and corpora. We advocate the use of relatively simple tags that do not require deep linguistic knowledge of the language but provide more structural information than POS tags and can be derived from automatically generated parse trees – a combination of properties that allows easy adoption of this model for new languages. We propose two fine-grain tagsets and evaluate our model using these tags, as well as POS tags and SuperARV tags in a speech recognition task and discuss future directions.

1 Introduction

In a number of language processing tasks, particularly automatic speech recognition (ASR) and machine translation (MT), there is the problem of selecting the best sequence of words from multiple hypotheses. This problem stems from the *noisy channel* approach to these applications. The noisy channel model states that the observed data, e.g., the acoustic signal, is the result of some input translated by some unknown stochastic process. Then the problem of finding the best sequence of words given the acoustic input, not approachable directly, is transformed into two separate models:

$$\underset{w_1^n}{argmax}\ p(w_1^n|A) = \underset{w_1^n}{argmax}\ p(A|w_1^n) \cdot p(w_1^n) \quad (1)$$

where A is the acoustic signal and w_1^n is a sequence of n words. $p(A|w_1^n)$ is called an acoustic model and $p(w_1^n)$ is the language model[1].

Typically, these applications use language models that compute the probability of a sequence in a generative way:

$$p(w_1^n) = \prod_{i=1}^{n} p(w_i|w_1^{i-1})$$

Approximation is required to keep the parameter space tractable. Most commonly the context is reduced to just a few immediately preceding words. This type of model is called an *ngram* model:

$$p(w_i|w_1^{i-1}) \approx p(w_i|w_{i-n+1}^{i-1})$$

Even with limited context, the parameter space can be quite sparse and requires sophisticated techniques for reliable probability estimation (Chen and Goodman, 1996). While the ngram models perform fairly well, they are only capable of capturing very shallow knowledge of the language.

There is extensive literature on a variety of methods that have been used to imbue models with syntactic and semantic information in different ways. These methods can be broadly categorized into two types:

- The first method uses surface words within its context, sometimes organizing them into deterministic classes. Models of this type include: (Brown et al., 1992; Zitouni, 2007), which use semantic word clustering, and (Bahl et al., 1990), which uses variable-length context.
- The other method adds stochastic variables to express the ambiguous nature of surface words[2]. To obtain the probability of the next

[1]Real applications use $argmax_{w_1^n}\ p(A|w_1^n) \cdot p(w_1^n)^\alpha \cdot n^\beta$ instead of Eq. 1, where α and β are set to optimize a heldout set.

[2]These variables have to be predicted by the model.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1114–1123,
Singapore, 6-7 August 2009.

word we need to sum over all assignments of the stochastic variables, as in Eq. 2.

$$p(w_i|w_1^{i-1}) = \sum_{t_1 \ldots t_i} p(w_i t_i|w_1^{i-1} t_1^{i-1}) \quad (2)$$
$$= \frac{\sum_{t_1 \ldots t_i} p(w_i t_i|w_1^{i-1} t_1^{i-1}) p(w_1^{i-1} t_1^{i-1})}{\sum_{t_1 \ldots t_{i-1}} p(w_1^{i-1} t_1^{i-1})}$$

Models of this type, which we call *joint* models since they essentially predict joint events of words and some random variable(s), include (Chelba and Jelinek, 2000) which used POS tags in combination with "parser instructions" for constructing a full parse tree in a left-to-right manner; (Wang et al., 2003) used SuperARVs (complex tuples of dependency information) without resolving the dependencies, thus called *almost parsing*; (Niesler and Woodland, 1996; Heeman, 1999) utilize part of speech (POS) tags. Note that some models reduce the context by making the following approximation:

$$p(w_i t_i|w_1^{i-1} t_1^{i-1}) \approx p(w_i|t_i) \cdot p(t_i|t_1^{i-1}) \quad (3)$$

thus, transforming the problem into a standard HMM application. However, these models perform poorly and have only been able to improve over the ngram model when interpolated with it (Niesler and Woodland, 1996).

Although joint models have the potential to better express variability in word usage through the introduction of additional latent variables, they do not necessarily perform better because the increased dimensionality of the context substantially increases the already complex problem of parameter estimation. The complexity of the space also makes computation of the probability a challenge because of space and time constraints. This makes the choice of the random variables a matter of utmost importance.

The model presented in this paper has some elements borrowed from prior work, notably (Heeman, 1999; Xu and Jelinek, 2004), while others are novel.

1.1 Paper Outline

The message we aim to deliver in this paper can be summarized in two theses:

- *Use fine-grain syntactic tags in a joint LM.* We propose a joint language model that can be used with a variety of tagsets. In Section 2, we describe those that we used in our experiments. Rather than tailoring our model to these tagsets, we aim for flexibility and propose an information theoretic framework for quick evaluation for tagsets, thus simplifying the creation of new tagsets. We show that our model with fine-grain tagsets outperform the coarser POS model, as well as the ngram baseline, in Section 5.

- *Address the challenges* that arise in a joint language model with fine-grain tags. While the idea of using joint language modeling is not novel (Chelba and Jelinek, 2000; Heeman, 1999), nor is the idea of using fine-grain tags (Bangalore, 1996; Wang et al., 2003), none of prior papers focus on the issues that arise from the combination of joint language modeling with fine-grain tags, both in terms of reliable parameter estimation and scalability in the face of the increased computational complexity. We dedicate Sections 3 and 4 to this problem.

In Section 6, we summarize conclusions and lay out directions for future work.

2 Structural Information

As we have mentioned, the selection of the random variable in Eq. 2 is extremely important for the performance of the model. On one hand, we would like for this variable to provide maximum information. On the other hand, as the number of parameters grow, we must address reliable parameter estimation in the face of sparsity, as well as increased computational complexity. In the following section we will compare the use of SuperARVs, POS tags, and other structural tags derived from parse trees.

2.1 POS Tags

Part-of-speech tags can be easily obtained for unannotated data using off-the-shelf POS taggers or PCFG parsers. However, the amount of information these tags typically provide is very limited,

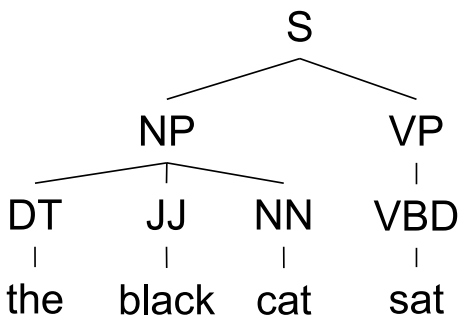


Figure 1: A parse tree example

e.g., while it is helpful to know whether *fly* is a verb or a noun, knowing that *you* is a personal pronoun does not carry the information whether it is a subject or an object (given the Penn Tree Bank tagset), which would certainly help to predict the following word.

2.2 SuperARV

The SuperARV essentially organizes information concerning one consistent set of dependency links for a word that can be directly derived from its syntactic parse. SuperARVs encode lexical information as well as syntactic and semantic constraints in a uniform representation that is much more fine-grained than POS. It is a four-tuple $(C; F; R+; D)$, where C is the lexical category of the word, F is a vector of lexical features for the word, $R+$ is a set of governor and need labels that indicate the function of the word in the sentence and the types of words it needs, and D represents the relative position of the word and its dependents. We refer the reader to the literature for further details on SuperARVs (Wang and Harper, 2002; Wang et al., 2003).

SuperARVs can be produced from parse trees by applying deterministic rules. In this work we use SuperARVs as individual tags and do not cluster them based of their structure. While SuperARVs are very attractive for language modeling, developing such a rich set of annotations for a new language would require a large amount of human effort.

We propose two other types of tags which have not been applied to this task, although similar information has been used in parsing.

2.3 Modifee Tag

This tag is a combination of the word's POS tag and the POS tag of its governor role. We designed it to resemble dependency parse structure. For example, the sentence in Figure 1 would be tagged: *the/DT-NN black/JJ-NN cat/NN-VBD sat/VBD-root*. Henceforth, we will refer to this kind of tag as *head*.

2.4 Parent Constituent

This tag is a combination of the word's POS tag with its immediate parent in the parse tree, along with the POS tag's relative position among its siblings. We refer to this type of tags as *parent*. The example in Figure 1 will be tagged: *the/DT-NP-start black/JJ-NP-mid cat/NN-NP-end sat/VB-VP-single*. This tagset is designed to represent constituency information.

Note that the *head* and *parent* tagsets are more language-independent (all they require is a treebank) than the SuperARVs which, not only utilized the treebank, but were explicitly designed by a linguist for English only.

2.5 Information Theoretic Comparison of Tags

As we have mentioned in Section 1, the choice of the tagset is very important to the performance of the model. There are two conflicting intuitions for tags: on one hand they should be specific enough to be helpful in the language model's task; on the other hand, they should be easy for the LM to predict.

Of course, in order to argue which tags are more suitable, we need some quantifiable metrics. We propose an information theoretic approach:

- To quantify how hard it is to predict a tag, we compute the conditional entropy:

$$\begin{aligned} H_p(t_i|w_i) &= H_p(t_i w_i) - H_p(w_i) \\ &= \sum_{w_i t_i} p(t_i w_i) \log p(t_i|w_i) \end{aligned}$$

- To measure how helpful a tagset is in the LM task, we compute the reduction of the conditional cross entropy:

$$\begin{aligned} & H_{\tilde{p},q}(w_i|w_{i-1}t_{i-1}) - H_{\tilde{p},q}(w_i|w_{i-1}) = \\ & \quad - \sum_{w_{i-1}^i t_{i-1}} \tilde{p}(w_{i-1}^i t_{i-1}) \log q(w_i|w_{i-1}t_{i-1}) \\ & \quad + \sum_{w_{i-1}^i} \tilde{p}(w_{i-1}^i) \log q(w_i|w_{i-1}) \\ = & \quad - \sum_{w_{i-1}^i t_{i-1}} \tilde{p}(w_{i-1}^i t_{i-1}) \log \frac{q(w_i|w_{i-1}t_{i-1})}{q(w_i|w_{i-1})} \end{aligned}$$

Note that in this case we use conditional *cross* entropy because conditional entropy has the tendency to overfit the data as we select more and more fine-grain tags. Indeed, $H_p(w_i|w_{i-1}t_{i-1})$ can be reduced to zero if the tags are specific enough, which would never happen in reality. This is not a problem for the former metric because the context there, w_i, is fixed. For this metric, we use a smoothed distribution $\tilde{p}$ computed on the training set[3] and the test distribution q.

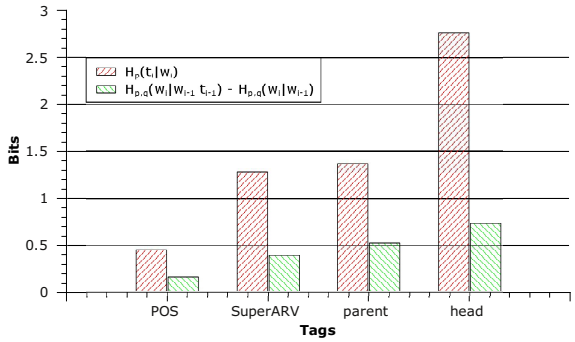


Figure 2: Changes in entropy for different tagsets

The results of these measurements are presented in Figure 2. POS tags, albeit easy to predict, provide very little additional information about the following word, and therefore we would not expect them to perform very well. The *parent* tagset seems to perform somewhat better than SuperARVs – it provides 0.13 bits more information while being only 0.09 bits harder to predict based on the word. The *head* tagset is interesting: it provides 0.2 bits more information about the following word (which would correspond to 15% perplexity reduction if we had perfect tags), but on the other hand the model is less likely to predict these tags accurately.

This approach is only a crude estimate (it uses only unigram and bigram context) but it is very useful for designing tagsets, e.g., for a new language, because it allows us to assess relative performance of tagsets without having to train a full model.

[3]We used *one-count* smoothing (Chen and Goodman, 1996).

3 Language Model Structure

The size and sparsity of the parameter space of the joint model necessitate the use of dimensionality reduction measures in order to make the model computationally tractable and to allow for accurate estimation of the model's parameters. We also want the model to be able to easily accommodate additional sources of information such as morphological features, prosody, etc. In the rest of this section, we discuss avenues we have taken to address these problems.

3.1 Decision Tree Clustering

Binary decision tree clustering has been shown to be effective for reducing the parameter space in language modeling (Bahl et al., 1990; Heeman, 1999) and other language processing applications, e.g., (Magerman, 1994). Like any clustering algorithm, it can be represented by a function H that maps the space of histories to a set of equivalence classes.

$$p(w_i t_i | w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) \approx p(w_i t_i | H(w_{i-n+1}^{i-1} t_{i-n+1}^{i-1})) \quad (4)$$

While the tree construction algorithm is fairly standard – to recursively select binary questions about the history optimizing some function – there are important decisions to make in terms of which questions to ask and which function to optimize. In the remainder of this section, we discuss the decisions we made regarding these issues.

3.2 Factors

The Factored Language Model (FLM) (Bilmes and Kirchhoff, 2003) offers a convenient view of the input data: it represents every word in a sentence as a tuple of factors. This allows us to extend the language model with additional parameters. In an FLM, however, all factors have to be deterministically computed in a joint model; whereas, we need to distinguish between the factors that are given or computed and the factors that the model must predict stochastically. We call these types of factors *overt* and *hidden*, respectively. Examples of overt factors include surface words, morphological features such as suffixes, case information when available, etc., and the hidden factors are POS, SuperARVs, or other tags.

Henceforth, we will use *word* to represent the set of overt factors and *tag* to represent the set of hidden factors.

3.3 Hidden Factors Tree

Similarly to (Heeman, 1999), we construct a binary tree where each tag is a leaf; we will refer to this tree as the *Hidden Factors Tree* (HFT). We use Minimum Discriminative Information (MDI) algorithm (Zitouni, 2007) to build the tree. The HFT represents a hierarchical clustering of the tag space. One of the reasons for doing this is to allow questions about subsets of tags rather than individual tags alone[4].

Unlike (Heeman, 1999), where the tree of tags was only used to create questions, this representation of the tag space is, in addition, a key feature of our decoding optimizations, which we discuss in Section 4.

3.4 Questions

The context space is partitioned by means of binary *questions*. We use different types of questions for hidden and overt factors.

- Questions about surface words are constructed using the Exchange algorithm (Martin et al., 1998). This algorithm takes the set of words that appear at a certain position in the training data associated with the current node in the history tree and divides the set into two complementary subsets greedily optimizing some target function (we use the average entropy of the marginalized word distribution, the same as for question selection). Note that since the algorithm only operates on the words that appear in the training data, we need to do something more to account for the unseen words. Thus, to represent this type of question, we create the history tree structure depicted in Fig. 4.

  For other overt factors with smaller vocabularies, such as suffixes, we use equality questions.

- As we mentioned in Section 3.3, we use the Hidden Factors Tree to create questions about hidden factors. Note that every node in a binary tree can be represented by a binary path from the root with all nodes under an inner node sharing the same prefix. Thus, a question about whether a tag belongs to a subset of tags dominated by a node can be expressed as whether the tag's path matches the binary prefix.

[4]Trying all possible subsets of tags is not feasible since there are $2^{|T|}$ of them. The tree allows us to reduce the number to $O(T)$ of the most meaningful (as per the clustering algorithm) subsets.

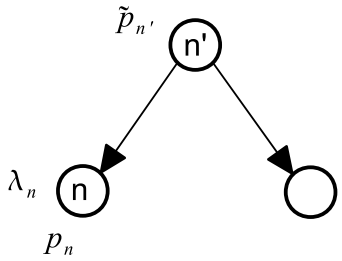


Figure 3: Recursive smoothing: $\tilde{p_n} = \lambda_n p_n + (1 - \lambda_n)\tilde{p_{n'}}$

3.5 Optimization Criterion and Stopping Rule

To select questions we use the average entropy of the marginalized word distribution. We found that this criterion significantly outperforms the entropy of the distribution of joint events. This is probably due to the increased sparsity of the joint distribution and the fact that our ultimate metrics, i.e., WER and word perplexity, involve only words.

3.6 Distribution Representation

In a cluster H_x, we factor the joint distribution as follows:

$$p(w_i t_i | H_x) = p(w_i | H_x) \cdot p(t_i | w_i, H_x)$$

where $p(t_i|w_i, H_x)$ is represented in the form of an HFT, in which each leaf has the probability of a tag and each internal node contains the sum of the probabilities of the tags it dominates. This representation is designed to assist the decoding process described in Section 4.

3.7 Smoothing

In order to estimate probability distributions at the leaves of the history tree, we use the following recursive formula:

$$\tilde{p}_n(w_i t_i) = \lambda_n p_n(w_i t_i) + (1 - \lambda_n)\tilde{p}_{n'}(w_i t_i) \quad (5)$$

where n' is the n-th node's parent, $p_n(w_i t_i)$ is the distribution at node n (see Figure 3). The

root of the tree is interpolated with the distribution $p_{unif}(w_i t_i) = \frac{1}{|V|} p_{ML}(t_i|w_i)$[5]. To estimate interpolation parameters λ_n, we use the EM algorithm described in (Magerman, 1994); however, rather than setting aside a separate development set of optimizing λ_n, we use 4-fold cross validation and take the geometric mean of the resulting coefficients[6]. We chose this approach because a small development set often does not overlap with the training set for low-count nodes, leading the EM algorithm to set $\lambda_n = 0$ for those nodes.

Let us consider one leaf of the history tree in isolation. Its context can be represented by the path to the root, i.e., the sequence of questions and answers $q_1, \ldots q_{(n')'} q_{n'}$ (with q_1 being the answer to the topmost question):

$$\tilde{p}_n(w_i t_i) = \tilde{p}(w_i t_i | q_1 \ldots q_{(n')'} q_{n'})$$

Represented this way, Eq. 5 is a variant of Jelinek-Mercer smoothing:

$$\begin{aligned}\tilde{p}(w_i t_i | q_1 \ldots q_{n'}) \;\; = \;\; & \lambda_n p(w_i t_i | q_1 \ldots q_{n'}) \; + \\ & (1 - \lambda_n) \tilde{p}(w_i t_i | q_1 \ldots q_{(n')'})\end{aligned}$$

For backoff nodes (see Fig. 4), we use a lower order model[7] interpolated with the distribution at the backoff node's grandparent (see node A in Fig. 4):

$$\begin{aligned}&\tilde{p}_B(w_i t_i | w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) = \\ &\quad \alpha_A \tilde{p}_{bo}(w_i t_i | w_{i-n+2}^{i-1} t_{i-n+2}^{i-1}) \; + \; (1 - \alpha_A) \tilde{p}_A(w_i t_i)\end{aligned}$$

How to compute α_A is an open question. For this study, we use a simple heuristic based on observation that the further node A is from the root the more reliable the distribution $\tilde{p}_A(w_i t_i)$ is, and hence α_A is lower. The formula we use is as follows:

$$\alpha_A = \frac{1}{\sqrt{1 + distanceToRoot(A)}}$$

[5]We use this distribution rather than uniform joint distribution $\frac{1}{|V||T|}$ because we do not want to allow word-tag pairs that have never been observed. The idea is similar to (Thede and Harper, 1999).

[6]To avoid a large number of zeros due to the product, we set a minimum for λ to be 10^{-7}.

[7]The lower order model is constructed by the same algorithm, although with smaller context. Note that the lower order model can back off on words or tags, or both. In this paper we backoff both on words and tags, i.e., $p(w_i t_i | w_{i-2}^{i-1} t_{i-2}^{i-1})$ backs off to $p(w_i t_i | w_{i-1} t_{i-1})$, which in turn backs off to the unigram $p(w_i t_i)$.

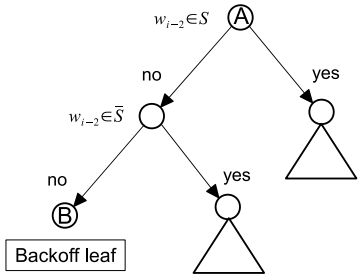


Figure 4: A fragment of the decision tree with a backoff node. $S \cup \bar{S}$ is the set of words observed in the training data at the node A. To account for unseen words, we add the backoff node B.

4 Decoding

As in HMM decoding, in order to compute probabilities for i-th step, we need to sum over $|T|^{n-1}$ possible combinations of tags in the history, where T is the set of tags and n is the order of the model. With $|T|$ predictions for the i-th step, we have $O(|T|^n)$ computational complexity per word. Straightforward computation of these probabilities is problematic even for a trigram model with POS tags, i.e., $n = 3, |T| \approx 40$. A standard approach to limit computational requirements is to use beam search where only $\mathcal{N}$ most likely paths are retained. However, with fine-grain tags where $|T| \approx 1,500$, a tractable beam size would only cover a small fraction of the whole space, leading to search errors such as pruning good paths.

Note that we have a history clustering function (Eq. 4) represented by the decision tree, and we should be able to exploit this clustering to eliminate unnecessary computations involving equivalent histories. Note that words in the history are known exactly, thus we can create a projection of the clustering function H in Eq. 4 to the plane $w_{i-n+1}^{i-1} = const$, i.e., where words in the context are fixed to be whatever is observed in the history:

$$H(w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) \;\Rightarrow\; \hat{H}_{w_{i-n+1}^{i-1} = const}(t_{i-n+1}^{i-1}) \tag{6}$$

The number of distinct clusters in the projection $\hat{H}$ depends on the decision tree configuration and can vary greatly for different words w_{i-n+1}^{i-1} in the history, but generally it is relatively small:

$$|\hat{H}_{w_{i-n+1}^{i-1} = const}(t_{i-n+1}^{i-1})| \;\ll\; |T^{n-1}| \tag{7}$$

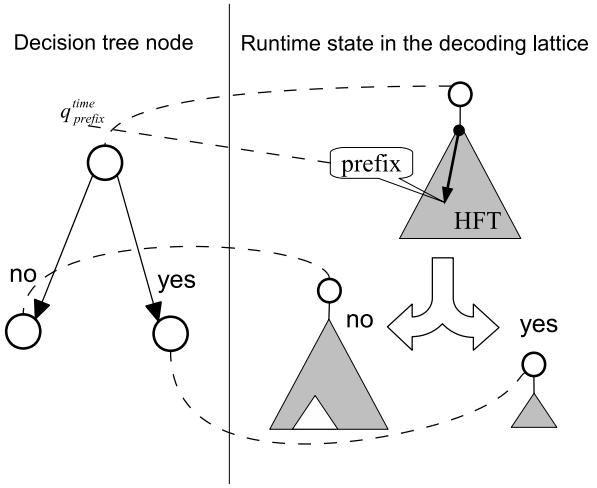


Figure 5: Questions about hidden factors split states (see Figure 6) in the decoding lattice represented by HFTs.

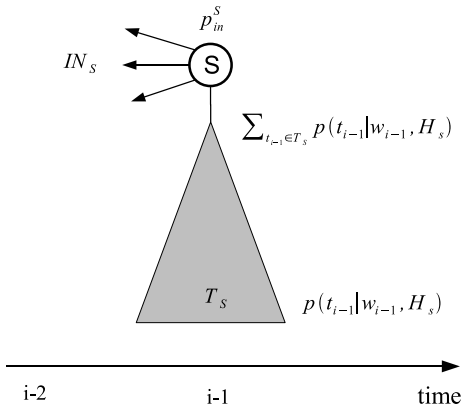


Figure 6: A state S in the decoding lattice. p_{in}^{S} is the probability of reaching the state S through the set of links IN_S. The probabilities of generating the tags $p(t_{i-1}|w_{i-1}, H_s), (t_{i-1} \in T_S)$ are represented in the form of the HFT.

thus, the number of probabilities that we need to compute is $|\hat{H}_{w_{i-n+1}^{i-1}=const}| \cdot |T|$.

Our decoding algorithm works similarly to HMM decoding with the exception that the set of hidden states is not predetermined. Let us illustrate how it works in the case of a bigram model. Recall that the set of tags T is represented as a binary tree (HFT) and the only type of questions about tags is about matching a binary prefix in the HFT. Such a question dissects the HFT into two parts as depicted in Figure 5. The cost of this operation is $O(\log |T|)$.

We represent states in the decoding lattice as shown in the Figure 6, where p_{in}^{S} is the probability of reaching the state S:

$$p_{in}^{S} = \sum_{S' \in IN_S} \left[p_{in}^{S'}\, p(w_{i-2}|H_{S'}) \sum_{t \in T_{S'}} p(t|w_{i-2}H_{S'}) \right]$$

where IN_S is the set of incoming links to the state S from the previous time index, and $T_{S'}$ is the set of tags generated from the state S' represented as a fragment of the HFT. Note, that since we maintain the property that the probability assigned to an inner node of the HFT is the sum of probabilities of the tags it dominates, the sum $\sum_{t \in T_{S'}} p(t|w_{i-2}H_{S'})$ is located at the root of $T_{S'}$, and therefore this is an $O(1)$ operation.

Now given the state S at time $i-1$, in order to generate tag predictions for i-th word, we apply questions from the history clustering tree, starting from the top. Questions about overt factors always follow either a *true* or *false* branch, implicitly computing the projection in Eq. 6. Questions about hidden factors, can split the state S into two states S_{true} and S_{false}, each retaining a part of T_S as shown in the Figure 5.

The process continues until each fragment of each state at the time $i-1$ reaches the bottom of the history tree, at which point new states for time i are generated from the clusters associated with leaves. The states at $i-1$ that generate the cluster $H_{\tilde{S}}$ become the incoming links to the state $\tilde{S}$.

Higher order models work similarly, except that at each time we consider a state S at time $i-1$ along with one of its incoming links (to some depth according to the size of the context).

5 Experimental Setup

To evaluate the impact of fine-grain tags on language modeling, we trained our model with five settings: In the first model, questions were restricted to be about overt factors only, thus making it a tree-based word model. In the second model, we used POS tags. To evaluate the effect of fine-grain tags, we train two models: *head* and *parent* described in Section 2.3 and Section 2.4 respectively. Since our joint model can be used with any kind of tags, we also trained it with SuperARV tags (Wang et al., 2003). The SuperARVs were created from the same parse trees that were used to produce POS and fine-grain tags. All our models, including SuperARV, use trigram context. We include standard trigram, four-gram, and five-

gram models for reference. The ngram models were trained using SRILM toolkit with interpolated modified Kneser-Ney smoothing.

We evaluate our model with an nbest rescoring task using 100-best lists from the DARPA WSJ'93 and WSJ'92 20k open vocabulary data sets. The details on the acoustic model used to produce the nbest lists can be found in (Wang and Harper, 2002). Since the data sets are small, we combined the 93et and 93dt sets for evaluation and used 92et for the optimization[8]. We transformed the nbest lists to match PTB tokenization, namely separating possessives from nouns, *n't* from auxiliary verbs in contractions, as well as contractions from personal pronouns.

All language models were trained on the NYT 1994-1995 section of the English Gigaword corpus (approximately 70M words). Since the New York Times covers a wider range of topics than the Wall Street Journal, we eliminated the most irrelevant stories based on their trigram coverage by sections 00-22 of WSJ. We also eliminated sentences over 120 words, because the parser's performance drops significantly on long sentences. After parsing the corpus, we deleted sentences that were assigned a very low probability by the parser. Overall we removed only a few percent of the data; however, we believe that such a rigorous approach to data cleaning is important for building discriminating models.

Parse trees were produced by an extended version of the Berkeley parser (Huang and Harper, 2009). We trained the parser on a combination of the BN and WSJ treebanks, preprocessed to make them more consistent with each other. We also modified the trees for the speech recognition task by replacing numbers and abbreviations with their verbalized forms. We pre-processed the NYT corpus in the same way, and parsed it. After that, we removed punctuation and downcased words. For the ngram model, we used text processed in the same way.

In *head* and *parent* models, tag vocabularies contain approximately 1,500 tags each, while the SuperARV model has approximately 1,400 distinct SuperARVs, most of which represent verbs (1,200).

In these experiments we did not use overt factors other than the surface word because they split <unk>, effectively changing the vocabulary thus making perplexity incomparable to models without these factors, without improving WER noticeably. However, we do plan to use more overt factors in Machine Translation experiments where a language model faces a wider range of OOV phenomena, such as abbreviations, foreign words, numbers, dates, time, etc.

Models	WER
trigram (baseline)	17.5
four-gram	17.7
five-gram	17.8
Word Tree	17.3
POS Tags	17.0
Head Tags	16.8
Parent Tags	**16.7**
SuperARV	16.9

Table 1: WER results, optimized on 92et set, evaluated on combined 93et and 93dt set. The Oracle WER is 9.5%.

Table 1 summarizes performance of the LMs on the rescoring task. The *parent* tags model outperforms the trigram baseline model by 0.8% WER. Note that four- and five-gram models fail to outperform the trigram baseline. We believe this is due to the sparsity as well as relatively short sentences in the test set (16 words on average).

Interestingly, whereas the improvement of the POS model over the baseline is not statistically significant ($p < 0.10$)[9], the fine-grain models outperform the baseline much more reliably: $p < 0.03$ (SuperARV) and $p < 0.007$ (*parent*).

We present perplexity evaluations in Table 2. The perplexity was computed on Section 23 of WSJ PTB, preprocessed as the rest of the data we used. The *head* model has the lowest perplexity outperforming the baseline by 9%. Note, it even outperforms the five-gram model, although by a small 2% margin.

Although the improvements by the fine-grain tagsets over POS are not significant (due to the small size of the test set), the reductions in perplexity suggest that the improvements are not random.

[8]We optimized the LM weight and computed WER with scripts in the SRILM and NIST SCTK toolkits.

[9]For statistical significance, we used SCTK implementation of the *mapsswe* test.

Models	PPL
trigram (baseline)	162
four-gram	152
five-gram	150
Word Tree	160
POS Tags	154
Head Tags	**147**
Parent Tags	150
SuperARV	150

Table 2: Perplexity results on Section 23 WSJ PTB

6 Conclusion and Future Work

In this paper, we presented a joint language modeling framework. Unlike any prior work known to us, it was not tailored for any specific tag set, rather it was designed to accommodate any set of tags, especially large sets ($\sim 1,000$), which present challenges one does not encounter with smaller tag sets, such at POS tags. We discussed these challenges and our solutions to them. Some of the solutions proposed are novel, particularly the decoding algorithm.

We also proposed two simple fine-grain tagsets, which, when applied in language modeling, perform comparably to highly sophisticated tag sets (SuperARV). We would like to stress that, while our fine-grain tags did not significantly outperform SuperARVs, the former use much less linguistic knowledge and can be automatically induced for any language with a treebank.

Because a joint language model inherently predicts hidden events (tags), it can also be used to generate the best sequence of those events, i.e., tagging. We evaluated our model in the POS tagging task and observed similar results: the fine-grain models outperform the POS model, while both outperform the state-of-the-art HMM POS taggers. We refer to (Filimonov and Harper, 2009) for details on these experiments.

We plan to investigate how parser accuracy and data selection strategies, e.g., based on parser confidence scores, impact the performance of our model. We also plan on evaluating the model's performance on other genres of speech, as well as in other tasks such as Machine Translation. We are also working on scaling our model further to accommodate amounts of data typical for modern large-scale ngram models. Finally, we plan to apply the technique to other languages with treebanks, such as Chinese and Arabic.

We intend to release the source code of our model within several months of this publication.

7 Acknowledgments

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023 and NSF IIS-0703859. Any opinions, findings and/or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the institutions where the work was completed.

References

Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. 1990. A tree-based statistical language model for natural language speech recognition. *Readings in speech recognition*, pages 507–514.

Srinivas Bangalore. 1996. 'Almost parsing' technique for language modeling. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 1173–1176.

Jeff A. Bilmes and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *Proceedings of HLT/NACCL, 2003*, pages 4–6.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling for speech recognition. *CoRR*.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318, Morristown, NJ, USA. Association for Computational Linguistics.

Denis Filimonov and Mary Harper. 2009. Measuring tagging performance of a joint language model. In *Proceedings of the Interspeech 2009*.

Peter A. Heeman. 1999. POS tags and decision trees for language modeling. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 129–137.

Zhongqiang Huang and Mary Harper. 2009. Self-Training PCFG grammars with latent annotations across languages. In *Proceedings of the EMNLP 2009*.

David M. Magerman. 1994. *Natural language parsing as statistical pattern recognition*. Ph.D. thesis, Stanford, CA, USA.

Sven Martin, Jorg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. In *Speech Communication*, pages 1253–1256.

Thomas R. Niesler and Phil C. Woodland. 1996. A variable-length category-based n-gram language model. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1:164–167 vol. 1, May.

Scott M. Thede and Mary P. Harper. 1999. A second-order hidden markov model for part-of-speech tagging. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 175–182.

Wen Wang and Mary P. Harper. 2002. The SuperARV language model: investigating the effectiveness of tightly integrating multiple knowledge sources. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 238–247, Morristown, NJ, USA. Association for Computational Linguistics.

Wen Wang, Mary P. Harper, and Andreas Stolcke. 2003. The robustness of an almost-parsing language model given errorful training data. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Peng Xu and Frederick Jelinek. 2004. Random forests in language modeling. In *in Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Imed Zitouni. 2007. Backoff hierarchical class n-gram language models: effectiveness to model unseen events in speech recognition. *Computer Speech & Language*, 21(1):88–104.

Bidirectional Phrase-based Statistical Machine Translation

Andrew Finch
NICT, Keihanna Science City,
Kyoto, 619-0288, Japan
andrew.finch@nict.go.jp

Eiichiro Sumita
NICT, Keihanna Science City,
Kyoto, 619-0288, Japan
eiichiro.sumita@nict.go.jp

Abstract

This paper investigates the effect of direction in phrase-based statistial machine translation decoding. We compare a typical phrase-based machine translation decoder using a left-to-right decoding strategy to a right-to-left decoder. We also investigate the effectiveness of a bidirectional decoding strategy that integrates both mono-directional approaches, with the aim of reducing the effects due to language specificity. Our experimental evaluation was extensive, based on 272 different language pairs, and gave the surprising result that for most of the language pairs, it was better decode from right-to-left than from left-to-right. As expected the relative performance of left-to-right and right-to-left strategies proved to be highly language dependent. The bidirectional approach outperformed the both the left-to-right strategy and the right-to-left strategy, showing consistent improvements that appeared to be unrelated to the specific languages used for translation. Bidirectional decoding gave rise to an improvement in performance over a left-to-right decoding strategy in terms of the BLEU score in 99% of our experiments.

1 Introduction

Human language production by its very nature is an ordered process. That is to say, words are written/uttered in a sequence. The current generation of phrase-based statistical machine translation (SMT) systems also generate their target word sequences according to an order. Since the generation process is symmetrical, there are two possible strategies that could be used to generate the target: from beginning to end; or from end to beginning. Generating the target in the 'wrong' direction (the opposite direction to the way in which humans do) is counter intuitive, and possibly as a result of this, SMT systems typically generate the target word sequence in the same order as human language production. However it is not necessarily the case that this is most effective strategy for all language pairs. In this paper we investigate the effect of direction in phrase-based SMT decoding.

For the purposes of this paper, we will refer to target word sequence generation that follows the same order as human language production as *forward* generation, and generation in the opposite direction to human language production as *reverse* generation. These are often referred "left-to-right" and "right-to-left" respectively in the literature, but we avoid this notation as many languages are naturally written from right-to-left.

In earlier work (Watanabe and Sumita, 2002), it was hypothesized that the optimal direction for decoding was dependent on the characteristics of the target language. Their results show that for Japanese to English translation a reverse decoding strategy was the most effective, whereas for English to Japanese translation, a forward decoding strategy proved superior. In addition they implemented a bidirectional decoder, but their results were mixed. For English to Japanese translation, decoding bidirectionally gives higher performance, but for Japanese to English translation they were unable to improve performance by decoding bidirectionally. Their experiments were performed using a decoder based on IBM Model 4 using the translation techniques developed at IBM (Brown et al., 1993).

This work is closely related to the techniques proposed in (Watanabe and Sumita, 2002), but in our case we decode within the framework of a phrase-based SMT system, rather than the IBM model. Our intention was to explore the effect of direction in decoding within the context of a more

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1124–1132,
Singapore, 6-7 August 2009.

contemporary machine translation paradigm, and to experiment with a broader range of languages. The underlying motivation for our studies however remains the same. Languages have considerably different structure, and certain grammatical constructs tend to occupy particular positions within sentences of the same language, but different positions across languages. These differences may make it easier to tackle the automatic translation of a sentence in a given language from a particular direction. Our approach differs in that the decoding process of a phrased-based decoder is quite different from that used by (Watanabe and Sumita, 2002) since decoding is done using larger units making the re-ordering process much simpler. In (Watanabe and Sumita, 2002) only one language pair is considered, for our experiments we extended this to include translation among 17 different languages including the Japanese and English pair used in (Watanabe and Sumita, 2002). We felt that it was important to consider as many languages as possible in this study, as intuition and evidence from the original study suggests that the effect of direction in decoding is likely to be strongly language dependent.

The next section briefly describes the mechanisms underlying phrase-based decoding. Then we explain the principles behind the forward, reverse and bidirectional decoding strategies used in our experiments. Section 3 presents the experiments we performed. Section 4 gives the results and some analysis. Finally in Section 5, we conclude and offer possible directions for future research.

2 Phrase-based Translation

For our experiments we use the phrase-based machine translation techniques described in (Koehn, 2004) and (Koehn et al., 2007), integrating our models within a log-linear framework (Och and Ney, 2002).

One of the advantages of a log-linear model is that it is possible to integrate a diverse set of features into the model. For the decoders used in the experiments in this paper, we included the following feature functions:

- An n-gram language model over the target word sequence
  - Ensures the target word sequence is a likely sequence of words in the target language
- A phrase translation model
  - Effects the segmentation of the source word sequence, and is also responsible for the transformation of source phrases into target phrases.
- A target word sequence length model
  - Controls the length of the target word sequence. This is usually a constant term added for each word in the translation hypothesis.
- A lexicalized distortion model
  - Influences the reordering of the translated source phrases in the target word sequence using lexical context on the boundaries of the phrases being reordered.

2.1 Decoding

In a phrase-based SMT decoder, the word sequence of the target language is typically generated in order in a forward manner. The words at the start of the translation are generated first, then the subsequent words, in order until the final word of the target word sequence is generated. As the process is phrase-based, the translation is generated in a phrase-by-phrase manner, rather word-by-word. The basic idea is to segment the source word sequence into subsequences (phrases), then translate each phrase individually, and finally compose the target word sequence by reordering the translations of the source phrases. This composition must occur in a particular order, such that target words are generated sequentially from the start (or end in the case of reverse decoding) of the sentence. The reason that the target needs to be generated sequentially is to allow an n-gram language model to be applied to the partial target word sequence at each step of the decoding process.

This process is illustrated in Figure 1. In the decoding for both forward and reverse decoders the source sentence is segmented into 2 phrases: ”where is” and ”the station” (although in this example the segmentation is the same for both decoding strategies, it is not necessarily the case since the search processes are different). In the forward decoding process, first the English phrase “the station” is translated into the Japanese phrase “eki wa”. Initially the target sequence consists

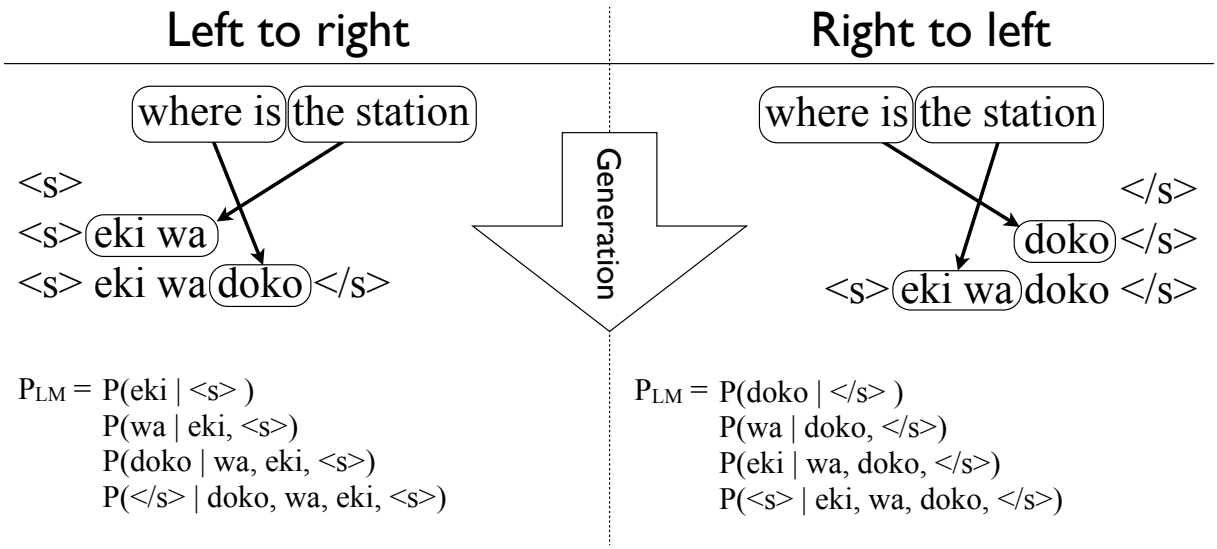


Figure 1: The phrase-based decoding process for an English to Japanese translation, in both forward and reverse directions. The n-gram language model probability calculation for the completed translation hypotheses are also shown on the bottom of the figure. See Section 2.1 for a description of the decoding process.

of only the start of sentence marker "$\langle$s$\rangle$". This marker only serves as context to indicate the start of the sequence for the benefit of the language model. The first target phrase is separated into its component words and each word is added in order to the target word sequence. Each addition causes an application of the language model, hence in Figure 1 the first term of P_{LM} is $P(\text{eki}|\langle\text{s}\rangle)$, the second is $P(\text{wa}|\langle\text{s}\rangle)$ and so on. For reverse decoding, the target sentence is generated starting from the end of sentence marker $\langle/\text{s}\rangle$ with the language model context being to the right of the current word. For the case of bidirectional decoding, the model probability for the hypothesis is a linear interpolation of the scores for both forward and reverse hypotheses.

2.2 Direction in Decoding

Direction in decoding influences both the models used by the decoder and the search process itself. The direction of decoding determines the order in which *target* words are generated, the source phrases being translated in any order, therefore it is likely to be features of the target language rather than those of the the source language that determine the effect that the decoding direction has on decoder performance.

2.2.1 The Language Model

The fundamental difference between the language models of a forward decoder and that of a reverse decoder is the direction in which the model looks for its context. The forward model looks back to the start of the sentence, whereas the reverse model looks forward to the end of the sentence.

2.2.2 The Search

Assuming a full search, a unigram language model and no limitations on reordering, the forward and reverse decoding processes are equivalent. When these constraints are lifted, as is the case in the experiments in this paper, the two search processes diverge and can give rise to hypotheses that are different in character.

The partial hypotheses from early in the search process for forward decoding represent hypotheses for the first few words of the target word sequence, whereas the early partial hypotheses of a reverse decoder hold the last few words. This has two consequences for the search. The first is that (assuming a beam search as used in our experiments), certain candidate word sequences in the early stages of the search might be outside the beam and be pruned. The consequence of this is that sentences that start with (or end with in the case of reverse decoding) the pruned word sequence will not be considered during the remainder of the search. The second is that word se-

quences in the partial hypotheses are used in the context of the models used in the subsequent decoding. Thus, correctly decoding the start (or end for reverse decoding) of the sentence will benefit the subsequent decoding process.

3 Experiments

3.1 Experimental Data

The experiments were conducted on all possible pairings among 17 languages. A key to the acronyms used for languages together with information about their respective characteristics is given in Table 1.

We used all of the first ATR Basic Travel Expression Corpus (BTEC1) (Kikui et al., 2003) for these experiments. This corpus contains the kind of expressions that one might expect to find in a phrase-book for travelers. The corpus is similar in character to the IWSLT06 Evaluation Campaign on Spoken Language Translation (Paul, 2006) J-E open track. The sentences are relatively short (see Table 1) with a simple structure and a fairly narrow range of vocabulary due to the limited domain.

The experiments were conducted on data that contained no case information, and also no punctuation (this was an arbitrary decision that we believe had no impact on the results).

We used a 1000 sentence development corpus for all experiments, and the corpus used for evaluation consisted of 5000 sentences with a single reference for each sentence.

3.2 Training

Each instance of the decoder is a standard phrase-based machine translation decoder that operates according to the same principles as the publicly available PHARAOH (Koehn, 2004) and MOSES (Koehn et al., 2007) SMT decoders. In these experiments 5-gram language models built with Witten-Bell smoothing were used along with a lexicalized distortion model. The system was trained in a standard manner, using a minimum error-rate training (MERT) procedure (Och, 2003) with respect to the BLEU score (Papineni et al., 2001) on held-out development data to optimize the log-linear model weights. For simplicity, the MERT procedure was performed on independently on the forward and reverse decoders for the bidirectional system, rather them attempting to tune the parameters for the full system.

3.3 Translation Engines

3.3.1 Forward

The forward decoding translation systems used in these experiments represent the baseline of our experiments. They consist of phrase-based, multi-stack, beam search decoders commonly used in the field.

3.3.2 Reverse

The reverse decoding translation systems used in these experiments were exactly the same as the forward decoding systems. The difference being the that word sequences in the training, development, and source side of the test corpora were reversed prior to training the systems. The final output of the reverse decoders was reordered in a post processing step before evaluation.

3.3.3 Bidirectional

The decoder used for the bidirectional decoding experiments was modified in order to be able to decode both forward and reverse in separate instances of the decoder. Models for decoding in forward and reverse directions are loaded, and two decoding instances created. Scores for hypotheses that share the same target word sequence from the two decoders were combined at the end of the decoding process linearly using equal interpolation weights. Hypotheses that were generated by only one of the component decoders were not pruned. The scores from these hypotheses only had a contribution from the decoder that was able to generate them, the contribution from the other decoder being zero.

3.4 Decoding Constraints

The experiments reported in this paper were conducted with loose constraints on the decoding as overconstraining the decoding process could lead to differences between unidirectional and bidirectional strategies. More specificially, the decoding was done with a beam width of 100, no beam thresholding and no constraints on the reordering process. Figure 2 shows the effect of varying the beam width (stack size) in the search for forward decoder of the English to Japanese translation experiment. At the beam width of 100 used in our experiments, the gains from doubling the beam with are small (0.07 BLEU percentage points).

It is also important to note that a future cost identical to that used in the MOSES decoder

Abbreviation	Language	#Words	Avg. sent length	Vocabulary	Order
ar	Arabic	806853	5.16	47093	SVO
da	Danish	806853	5.16	47093	SVO
de	German	907354	5.80	23443	SVO
en	English	970252	6.21	12900	SVO
es	Spanish	881709	5.64	18128	SVO
fr	French	983402	6.29	17311	SVO
id	Indonesian (Malay)	865572	5.54	15527	SVO
it	Italian	865572	5.54	15527	SVO
ja	Japanese	1149065	7.35	15405	SOV
ko	Korean	1091874	6.98	17015	SOV
ms	Malaysian (Malay)	873959	5.59	16182	SVO
nl	Dutch	927861	5.94	19775	SVO
pt	Portuguese	881428	5.64	18217	SVO
ru	Russian	781848	5.00	32199	SVO
th	Thai	1211690	7.75	6921	SVO
vi	Vietnamese	1223341	7.83	8055	SVO
zh	Chinese	873375	5.59	14854	SVO

Table 1: Key to the languages, corpus statistics and word order. SVO denotes a language that predominantly has subject-verb-object order, and SOV denotes a language that predominantly has subject-object-verb order

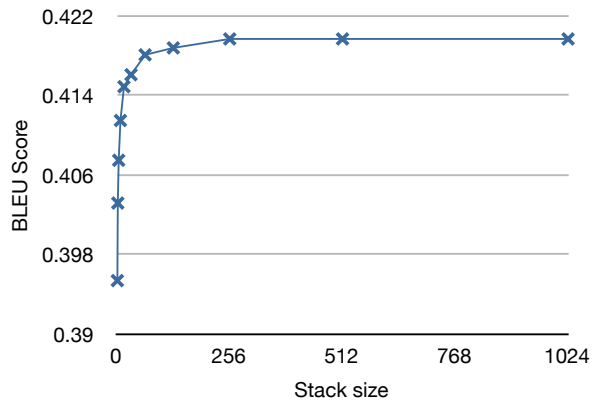


Figure 2: The performance of a forward decoder (En-Ja) with increasing stack size.

(Koehn et al., 2007) was also included in the scores for partial hypothesis during the decoding.

3.5 Computational Overhead

In the current implementation, bidirectional decoding takes twice as long as a mono-directional system. However, in a multi-threaded environment, each instance of the decoder is able to run on its own thread in parallel, and so this slowdown can be mitigated in some circumstances. Future generations of the bidirectional decoder will more tightly couple the two decoders, and we believe this will lead to faster and more effective search.

3.6 Evaluation

The results presented in this paper are given in terms of the BLEU score (Papineni et al., 2001). This metric measures the geometric mean of n-gram precision of n-grams drawn from the output translation and a set of reference translations for that translation.

There are large number of proposed methods for carrying out machine translation evaluation. Methods differ in their focus of characteristics of the translation (for example fluency or adequacy), and moreover anomolous results can occur if a single metric is relied on. Therefore, we also carried out evaluations using the NIST (Doddington, 2002), METEOR (Banerjee and Lavie, 2005), WER (Hunt, 1989), PER (Tillmann et al., 1997) and TER (Snover et al., 2005) machine translation evaluation techniques.

4 Results

The results of the experiments in terms of the BLEU score are given in Tables **??**, 5, 3 and 3. These results show the performance of the reverse and bidirectional decoding strategies relative to the usual forward decoding strategy. The cells in the tables that represent experiments in which

	ar	da	de	en	es	fr	id	it	ja	ko	ms	nl	pt	ru	th	vi	zh
ar	-	47.8	48.8	51.7	48.8	47.3	46.5	49.2	29.8	27.8	46.9	49.0	49.0	47.8	39.7	43.0	27.8
da	58.3	-	58.7	63.0	58.6	55.7	53.5	58.5	37.5	35.1	54.4	59.6	59.0	55.4	48.1	51.7	35.2
de	53.8	55.5	-	59.4	55.9	51.9	50.3	55.3	34.2	32.0	50.8	57.0	55.9	51.2	45.7	48.9	32.7
en	63.6	65.8	64.8	-	67.0	61.0	58.4	65.8	41.1	38.7	59.1	67.6	66.7	58.7	52.8	57.7	38.6
es	57.6	58.2	58.0	65.6	-	56.6	54.2	61.1	38.3	36.4	54.3	59.6	62.6	55.1	47.6	51.3	36.0
fr	57.8	58.3	58.0	62.3	58.9	-	52.7	57.4	39.1	37.7	53.8	58.3	57.9	54.8	47.7	50.4	37.6
id	54.7	52.8	52.8	56.6	53.7	51.0	-	53.1	37.2	35.6	86.4	53.8	53.0	51.3	46.4	48.4	34.9
it	54.1	53.4	54.4	59.4	56.4	51.8	49.2	-	34.4	32.8	49.9	55.1	56.2	50.5	44.0	47.0	33.6
ja	38.2	39.2	38.6	41.9	39.9	40.2	40.7	39.5	-	69.4	40.4	39.5	39.7	37.8	37.3	37.2	52.1
ko	34.4	35.3	34.6	38.2	36.3	36.2	36.8	35.6	66.4	-	36.6	35.6	36.3	34.5	34.2	34.1	46.4
ms	54.5	52.7	52.6	56.2	53.4	50.6	82.5	53.2	36.8	34.9	-	53.6	53.4	51.3	46.7	49.2	34.8
nl	55.1	57.3	58.8	63.2	58.5	54.5	52.4	57.1	36.7	34.1	53.4	-	58.3	53.5	48.7	50.7	35.2
pt	56.8	57.7	57.6	63.8	62.0	55.5	52.7	59.7	37.8	36.4	53.4	58.7	-	54.2	47.1	50.6	35.8
ru	51.4	49.1	50.2	53.3	52.0	48.7	48.6	51.6	31.9	29.5	49.1	50.9	50.5	-	41.8	43.7	30.0
th	53.8	55.0	54.8	58.2	55.8	53.3	55.0	54.8	41.4	39.2	55.4	55.9	55.5	53.0	-	56.0	40.4
vi	53.6	53.6	54.2	57.4	54.2	51.4	52.3	53.3	37.6	35.8	53.3	54.6	54.4	51.7	50.3	-	36.2
zh	32.0	33.0	32.6	34.6	33.2	33.7	34.2	33.2	47.8	43.5	33.9	33.4	32.6	32.2	31.1	29.7	-

Table 2: Baseline BLEU scores for all systems. The figures represent the scores in BLEU percentage points of the baseline left-to-right decoding systems. Source languages are indicated by the column headers, the row headers denoting the target languages.

the forward strategy outperformed the contrasting strategy are shaded in gray. The numbers in the cells represent the difference in BLEU percentage points for the systems being compared in that cell.

It is clear from Table 3 that for most of the language pairs (67% of them for BLEU, and a similar percentage for all the other metrics except METEOR), better evaluation scores were achieved by using a reverse decoding strategy than a forward strategy. This is a surprising result because language is produced naturally in a forward manner (by definition), and therefore one might expect this to also be the optimal direction for word sequence generation in decoding.

4.1 Word Order Typography

Following (Watanabe and Sumita, 2002), to explain the effects we observe in our results we look to the word order typography of the target language (Comrie and Vogel, 2000). The word order of a language is defined in terms of the order in which you would expect to encounter the finite verb (V) and its arguments, subject (S) and object (O). In most languages S precedes O and V. Whether or not O precedes or follows V defines the two most prevalent word order types SOV and SVO (Comrie and Vogel, 2000).

Two of the target languages in this study (Japanese and Korean) have the SOV word type, the remainder having the SVO word order type. In Table 3 looking at the rows for *ja* and *ko* we can see that for both of these languages reverse decoding outperformed forward decoding in only 4 out of 12 experiments. Furthermore these two languages were the two languages that benefited the most (in terms of the number of experimental cases) from forward decoding. The two languages also agree on the best decoding direction for 12 of the 16 language pairs. This apparent correlation may reflect similarities between the two languages (word order type, or other common features of the languages).

Given this evidence, it seems plausible that word order does account in part for the differences in performance when decoding in differing directions, but this can only be part of the explanation since there are 4 source languages for which reverse decoding yielded higher performance.

It should be noted that our results differ from those of (Watanabe and Sumita, 2002) for English to Japanese translation, who observed gains when decoding in the reverse direction for this language pair. It is hard to compare our results directly with theirs however, due to the differences in the decoders used in the experiments (ours being phrase-based, and theirs based on the IBM ap-

	ar	da	de	en	es	fr	id	it	ja	ko	ms	nl	pt	ru	th	vi	zh
ar	-	0.87	0.34	1.30	0.93	1.63	0.66	0.58	0.12	0.36	0.85	0.33	0.88	0.22	1.33	1.04	0.88
da	0.25	-	0.41	0.71	0.56	0.70	1.10	0.31	0.46	0.07	0.96	0.13	0.62	0.17	1.28	0.71	0.29
de	0.41	0.04	-	0.38	0.52	0.15	0.80	0.01	0.47	0.72	0.60	0.25	0.21	0.05	0.47	0.68	0.20
en	0.04	0.05	0.21	-	0.05	0.13	0.58	0.02	0.73	0.35	0.39	0.07	0.52	0.05	0.67	0.63	0.29
es	0.14	0.19	0.05	0.35	-	0.68	0.01	0.08	0.25	0.31	0.25	0.25	0.17	0.07	0.43	0.44	0.78
fr	0.37	0.57	0.38	0.66	0.21	-	0.36	0.28	0.15	0.45	0.22	0.46	0.64	0.10	0.25	0.58	0.31
id	0.16	0.02	0.31	1.45	0.58	0.50	-	0.34	0.03	0.27	0.00	0.42	0.57	0.36	0.53	1.04	0.59
it	0.28	0.72	0.36	0.27	0.08	0.30	0.11	-	0.07	0.12	0.37	0.23	0.05	0.37	0.04	0.63	0.37
ja	0.36	0.22	0.03	0.03	0.22	0.13	0.64	0.36	-	0.21	0.57	0.46	0.08	0.33	0.08	0.83	0.70
ko	0.35	0.01	0.31	0.03	0.12	0.07	0.13	0.21	0.42	-	0.29	0.07	0.42	0.40	0.44	0.62	0.05
ms	0.06	0.49	0.53	1.38	0.99	0.71	0.47	0.34	0.11	0.32	-	0.62	0.27	0.10	0.83	0.99	0.11
nl	0.26	0.03	0.26	0.30	0.20	0.19	0.47	0.23	0.13	0.06	0.06	-	0.08	0.09	0.06	1.00	0.15
pt	0.03	0.34	0.06	0.51	0.07	0.17	0.06	0.18	0.13	0.65	0.08	0.10	-	0.06	0.09	0.85	0.35
ru	0.25	0.58	0.67	0.74	0.01	0.48	0.50	0.27	0.41	0.38	0.13	0.38	0.46	-	0.88	0.56	0.49
th	0.19	0.28	0.21	0.41	0.05	0.23	0.30	0.00	0.34	0.04	0.25	0.07	0.21	0.08	-	0.46	0.25
vi	0.21	0.34	0.24	0.65	0.72	0.34	0.06	0.59	0.24	0.22	0.19	0.12	0.11	0.18	0.63	-	0.15
zh	0.43	0.26	0.42	0.05	0.15	0.31	0.16	0.28	0.00	0.31	0.40	0.14	0.67	0.18	0.39	0.21	-

Table 3: Gains in BLEU score from reverse decoding over a forward decoding strategy The numbers in the cells are the differences in BLEU percentage points between the systems. Shaded cells indicate the cases where forward decoding give a higher score. Source languages are indicated by the column headers, the row headers denoting the target languages.

Metric	Bi>For	Bi>Rev	Rev>For
BLEU	98.90	84.93	67.65
NIST	98.53	78.31	75.00
METEOR	99.63	95.96	50.74
WER	99.26	92.85	66.18
PER	98.53	84.97	70.59
TER	99.63	91.18	68.75

Table 4: Summary of the results using several automatic metrics for evaluation. Numbers in the table correspond to the percentage of experiments in which the condition at the head of the column was true (for example figure in the first row and first column means that for 98.9 percent of the language pairs the BLEU score for the bidirectional decoder was better than that of the forward decoder)

proach (Brown et al., 1993)).

The results were the similar in character when other MT evaluation methods were used. These results are summarized in Table 3.

4.2 Bidirectional Decoding

Table 5 shows the performance of the bidirectional decoder relative to a forward decoder. As can be seen from the table, in 269 out of the 272 experiments the bidirectional decoder outperformed the unidirectional decoder. The gains ranged from a maximum of 1.81 BLEU (translating from Thai to Arabic) points, to a minimum of -0.04 BLEU points (translating from Indonesian to Japanese) with the average gain over all experiments being 0.56 BLEU points. It is clear from our experiments that there is much to be gained from decoding bidirectionally. Our results were almost unanimously positive, and in all three negative cases the drop in performance was small.

5 Conclusion

In this paper we have investigated the effects on phrase-based machine translation performance of three different decoding strategies: forward, reverse and bidirectional. The experiments were conducted on a large set of source and target languages consisting of 272 experiments representing all possible pairings from a set of 17 languages. These languages were very diverse in character and included a broad selection of European and Asian languages. The experimental results revealed that for SVO word order languages it is usually better to decode in a reverse manner, and in contrast, for SOV word order languages it is usu-

	ar	da	de	en	es	fr	id	it	ja	ko	ms	nl	pt	ru	th	vi	zh
ar	-	0.66	0.51	1.03	0.65	0.75	0.59	0.47	0.46	0.85	0.59	0.69	0.39	0.30	1.81	1.30	0.85
da	0.27	-	0.61	0.63	0.38	0.60	0.59	0.29	1.04	0.79	0.69	0.45	0.89	0.27	1.28	0.87	0.47
de	0.52	0.51	-	0.54	0.44	0.42	0.70	0.40	0.74	0.45	0.83	0.37	0.28	0.34	0.77	0.90	0.84
en	0.53	0.01	0.32	-	0.23	0.25	0.56	0.19	1.11	0.59	0.28	0.27	0.45	0.60	0.89	0.61	0.58
es	0.28	0.48	0.45	0.56	-	0.43	0.12	0.26	0.57	0.64	0.56	0.06	0.04	0.24	1.16	1.23	0.68
fr	0.70	0.33	0.54	0.66	0.46	-	0.49	0.57	0.24	0.13	0.11	0.43	0.33	0.55	0.91	1.09	0.57
id	0.24	0.32	0.36	0.93	0.70	0.65	-	0.35	0.75	0.77	0.11	0.46	0.69	0.57	0.99	0.85	0.47
it	0.13	0.55	0.32	0.43	0.47	0.51	0.64	-	0.65	0.42	0.77	0.51	0.51	0.69	0.85	0.98	0.58
ja	0.38	0.62	0.60	0.61	0.38	0.73	0.04	0.43	-	0.35	0.05	0.70	0.30	0.38	0.53	0.17	0.02
ko	0.49	0.62	0.90	0.40	0.34	0.57	0.47	0.47	0.02	-	0.23	0.52	0.20	0.83	0.70	0.44	0.83
ms	0.37	0.57	0.63	0.92	0.81	0.75	0.36	0.54	0.70	1.31	-	0.76	0.35	0.51	1.14	0.70	0.35
nl	0.35	0.14	0.54	0.33	0.30	0.46	0.68	0.69	0.77	0.63	0.44	-	0.42	0.67	0.71	1.13	0.55
pt	0.46	0.21	0.37	0.21	0.17	0.49	0.47	0.24	0.88	0.45	0.54	0.39	-	0.41	0.94	1.15	0.90
ru	0.69	0.63	0.69	0.77	0.26	0.50	0.79	0.52	0.69	0.90	0.66	0.69	0.40	-	1.19	1.23	0.47
th	0.90	0.49	0.53	0.77	0.64	0.38	0.21	0.60	0.37	0.96	0.38	0.63	0.68	0.72	-	0.33	0.45
vi	0.64	0.61	0.42	1.09	0.84	0.63	0.34	0.70	0.59	0.39	0.16	0.56	0.36	0.50	0.77	-	0.53
zh	0.23	0.48	0.96	0.33	0.49	0.32	0.27	0.43	0.43	0.69	0.31	0.97	0.85	0.23	0.40	0.50	-

Table 5: Gains in BLEU score from decoding bidirectionally over a forward decoding strategy. The numbers in the cells are the differences in BLEU percentage points between the systems. Shaded cells indicate the cases where forward decoding gave a higher score. Source languages are indicated by the column headers, the row headers denoting the target languages.

ally better to decode in a forward direction. Our main contribution has been to show that a bidirectional decoding strategy is superior to both monodirectional decoding strategies. It might be argued that the gains arise simply from system combination. However, our systems are combined in a simple linear fashion, and gains will only arise when the second system contributes novel and useful information to into the combination. Furthermore, our systems are trained on two copies of the same data, no additional data is required. The gains from decoding bidirectionally were obtained very consistently, with only loose constraints on the decoding. This can be seen clearly in Table 5 where the results are almost unanimously positive. Moreover, these gains appear to be independent of the linguistic characteristics of the source and target languages.

In the future we would like to explore the possibilities created by more tightly coupling the forward and reverse components of the bidirectional decoder. Scores from partial hypotheses of both processes could be combined and used at each step of the decoding, making the search more informed. Furthermore, forward partial hypotheses and reverse hypotheses would 'meet' during decoding (when one decoding direction has covered words in the source that the other has yet to cover), and provide paths for each other to a final state in the search.

Acknowledgment

This work is partly supported by the Grant-in-Aid for Scientific Research (C) Number 19500137 and "Construction of speech translation foundation aiming to overcome the barrier between Asian languages", the Special Coordination Funds for Promoting Science and Technology of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: an automatic metric for mt evaluation with improved correlation with human judgments. In *ACL-2005: Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

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

Bernard Comrie and Petra M Vogel, editors. 2000. *Approaches to the Typography of Word Classes*. Mouton de Gruyter, Berlin.

G. Doddington. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. In *Proceedings of the HLT Conference*, San Diego, California.

Melvyn J. Hunt. 1989. Figures of merit for assessing connected-word recognisers. In *In Proceedings of the ESCA Tutorial and Research Workshop on Speech Input/Output Assessment and Speech Databases*, pages 127–131.

G. Kikui, E. Sumita, T. Takezawa, and S. Yamamoto. 2003. Creating corpora for speech-to-speech translation. In *Proceedings of EUROSPEECH-03*, pages 381–384.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowa, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *ACL 2007: proceedings of demo and poster sessions*, pages 177–180, Prague, Czeck Republic, June.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Machine translation: from real users to research: 6th conference of AMTA*, pages 115–124, Washington, DC.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 295–302.

Franz J. Och. 2003. Minimum error rate training for statistical machine trainslation. In *Proceedings of the ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Michael Paul. 2006. Overview of the iwslt 2006 evaluation campaign. In *Proceedings of the IWLST*.

Mathew Snover, Bonnie Dorr, Richard Schwartz, John Makhoul, Linnea Micciula, and Ralph Weischedel. 2005. A study of translation error rate with targeted human annotation. Technical report, University of Maryland, College Park and BBN Technologies, July.

C. Tillmann, S. Vogel, H. Ney, A. Zubiaga, and H. Sawaf. 1997. Accelerated dp based search for statistical translation. In *In European Conf. on Speech Communication and Technology*, pages 2667–2670.

Taro Watanabe and Eiichiro Sumita. 2002. Bidirectional decoding for statistical machine translation. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

Real-time decision detection in multi-party dialogue

Matthew Frampton, Jia Huang, Trung Huu Bui and **Stanley Peters**
Center for the Study of Language and Information
Stanford University
Stanford, CA, 94305, USA
{frampton|jiahuang|thbui|peters}@stanford.edu

Abstract

We describe a process for automatically detecting decision-making sub-dialogues in multi-party, human-human meetings in real-time. Our basic approach to decision detection involves distinguishing between different utterance types based on the roles that they play in the formulation of a decision. In this paper, we describe how this approach can be implemented in real-time, and show that the resulting system's performance compares well with other detectors, including an off-line version.

1 Introduction

In collaborative and organized work environments, people share information and make decisions through multi-party conversations, commonly referred to as meetings. The demand for automatic methods that process, understand and summarize information contained in audio and video recordings of meetings is growing rapidly, as evidenced by on-going projects which are focused on this goal, (Waibel et al., 2003; Janin et al., 2004). Our research is part of a general effort to develop a system that can automatically extract and summarize information such as conversational topics, action items, and decisions.

This paper concerns the development of a real-time decision detector — a system which can detect and summarize decisions as they are made during a meeting. Such a system could provide a summary of all of the decisions which have been made up until the current point in the meeting, and this is something which we expect will help users to enjoy more productive meetings. Certainly, good decision-making relies on access to relevant information, and decisions made earlier in a meeting often have a bearing on the current topic of discussion, and so form part of this relevant information. However, in a long and winding meeting, participants might not have these earlier decisions at the forefront of their minds, and so an accurate and succinct reminder, as provided by a real-time decision detector, could potentially be very useful. A record of earlier decisions could also help users to identify outstanding issues for discussion, and to therefore make better use of the remainder of the meeting.

Our approach to decision detection uses an annotation scheme which distinguishes between different types of utterance based on the roles which they play in the decision-making process. Such a scheme facilitates the detection of decision discussions (Fernández et al., 2008), and by indicating which utterances contain particular types of information, it also aids their summarization. To automatically detect decision discussions, we use what we refer to as *hierarchical classification*. Here, independent binary *sub-classifiers* detect the different decision dialogue acts, and then based on the sub-classifier hypotheses, a *super-classifier* determines which dialogue regions are decision discussions. In this paper then, we address the challenges for applying this approach in real-time, and produce a system which is able to detect decisions soon after they are made, (for example within a minute). We conduct tests and compare this system's performance with other detectors, including an off-line equivalent.

The remainder of the paper proceeds as follows. Section 2 describes related work, and Section 3 describes our annotation scheme for decision discussions, and our experimental data. Next, Section 4 explains the hierarchical classification approach in more detail, and Section 5 considers how it can be applied in real-time. Section 6 describes the experiments in which we test the real-time detector, and finally, Section 7 presents conclusions and ideas for future work.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1133–1141,
Singapore, 6-7 August 2009.

2 Related Work

Decisions are one of the most important meeting outputs. User studies (Lisowska et al., 2004; Banerjee et al., 2005) have confirmed that meeting participants consider this to be the case, and Whittaker et al. (2006) found that the development of an automatic decision detection component is critical to the re-use of meeting archives. As a result, with the new availability of substantial meeting corpora such as the ISL (Burger et al., 2002), ICSI (Janin et al., 2004) and AMI (McCowan et al., 2005) Meeting Corpora, recent years have seen an increasing amount of research on decision-making dialogue.

This recent research has tackled issues such as the automatic detection of agreement and disagreement (Hillard et al., 2003; Galley et al., 2004), and of the level of involvement of conversational participants (Wrede and Shriberg, 2003; Gatica-Perez et al., 2005). In addition, Verbree et al. (2006) created an argumentation scheme intended to support automatic production of argument structure diagrams from decision-oriented meeting transcripts. Only very recent research has specifically investigated the automatic detection of decisions, namely (Hsueh and Moore, 2007) and (Fernández et al., 2008).

Hsueh and Moore (2007) used the AMI Meeting Corpus, and attempted to automatically identify dialogue acts (DAs) in meeting transcripts which are "decision-related". Within any meeting, the authors decided which DAs were decision-related based on two different kinds of manually created summary: the first was an extractive summary of the whole meeting, and the second, an abstractive summary of the decisions which were made. Those DAs in the extractive summary which support any of the decisions in the abstractive summary were manually tagged as decision-related. Hsueh and Moore (2007) then trained a Maximum Entropy classifier to recognize this single DA class, using a variety of lexical, prosodic, dialogue act and conversational topic features. They achieved an F-score of 0.35, which gives an indication of the difficulty of this task.

Unlike Hsueh and Moore (2007), Fernández et al. (2008) made an attempt at modelling the structure of decision-making dialogue. They designed an annotation scheme that takes account of the different roles which different utterances play in the decision-making process — for example, their scheme distinguishes between decision DAs (DDAs) which initiate a discussion by raising a topic/issue, those which propose a resolution, and those which express agreement for a proposed resolution and cause it to be accepted as a decision. The authors applied the annotation scheme to a portion of the AMI corpus, and then took what they refer to as a *hierarchical classification* approach in order to automatically identify decision discussions and their component DAs. Here, one binary *Support Vector Machine (SVM)* per DDA class hypothesized occurrences of that DDA class, and then based on the hypotheses of these so-called *sub-classifiers*, a *super-classifier*, (a further SVM), determined which regions of dialogue represented decision discussions. This approach produced better results than the kind of "flat classification" approach pursued by Hsueh and Moore (2007) where a single classifier looks for examples of a single decision-related DA class. Using manual transcripts, and a variety of lexical, utterance, speaker, DA and prosodic features for the sub-classifiers, the super-classifier's F1-score was 0.58 according to a lenient match metric. Note that (Purver et al., 2007) had previously pursued the same basic approach as Fernández et al. (2008) in order to detect action items.

While both Hsueh and Moore (2007), and Fernández et al. (2008) attempted off-line decision detection, in this paper, we attempt real-time decision detection. We take the same basic approach as Fernández et al. (2008), and make changes to its implementation so that it can work effectively in real-time.

3 Data

The AMI corpus (McCowan et al., 2005), is a freely available corpus of multi-party meetings containing both audio and video recordings, as well as a wide range of annotated information including dialogue acts and topic segmentation. Conversations are all in English, but participants can include non-native English speakers. All of the meetings in our sub-corpus last around 30 minutes, and are scenario-driven, wherein four participants play different roles in a company's design team: *project manager*, *marketing expert*, *interface designer* and *industrial designer*. The discussions concern how to design a remote control.

We used the off-line version of the Decipher speech recognition engine (Stolcke et al., 2008) in

order to obtain off-line ASR transcripts for these 17 meetings, and the real-time version, to obtain real-time ASR transcripts. Decipher generates the transcripts by first producing Word Confusion Networks (WCNs) and then extracting their best paths. The real-time recognizer generates "live" transcripts with 5 to 15 seconds of latency for immediate display. In processing completed meetings, the off-line system makes seven recognition passes, including acoustic adaptation and language model rescoring, in about 4.2 times real-time (on a 4-score 2.6 GHz Opteron server). In general usage with multi-party dialogue, the word error rate (WER) for the off-line version of Decipher is approximately 23%, and for the real-time version, approximately 35%[1]. Stolcke et al. (2008) report a WER of 26.9% for the off-line version on AMI meetings.

The real-time ASR transcripts for the 17 meetings contain a total of 8440 utterances/dialogue acts, (around 496 per meeting), and the off-line ASR transcripts, 7495 utterances/dialogue acts, (around 441 per meeting).

3.1 Modelling Decision Discussions

We use the same annotation scheme as (Fernández et al., 2008) in order to model decision-making dialogue. As stated in Section 2, this scheme distinguishes between a small number of dialogue act types based on the role which they perform in the formulation of a decision. Recall that using this scheme in conjunction with *hierarchical classification* produced better decision detection than a "flat classification" approach with a single "decision-related" DA class. Since it indicates which utterances contain particular types of information, such a scheme also aids the summarization of decision discussions.

The annotation scheme (see Table 1 for a summary) is based on the observation that a decision discussion contains the following main structural components: (a) a topic or issue requiring resolution is raised, (b) one or more possible resolutions are considered, (c) a particular resolution is agreed upon, that is, it becomes the decision. Hence the scheme distinguishes between three corresponding decision dialogue act (DDA) classes: *Issue* (*I*), *Resolution* (*R*), and *Agreement* (*A*). Class *R* is further subdivided into *Resolution Proposal* (*RP*) and *Resolution Restatement* (*RR*). Note that an utterance can be assigned to more than one of these DDA classes, and that within a decision discussion, more than one utterance may correspond to a particular DDA class.

Here we use the sample decision discussion below in 1 in order to provide examples of the different DDA types. *I* utterances introduce the topic of the decision discussion, examples being *"Are we going to have a backup?"* and *"But would a backup really be necessary?"* On the other hand, *R* utterances specify the resolution which is ultimately adopted as the decision. *RP* utterances propose this resolution (e.g. *"I think maybe we could just go for the kinetic energy..."*), while *RR* utterances close the discussion by confirming/summarizing the decision (e.g. *"Okay, fully kinetic energy"*). Finally, *A* utterances agree with the proposed resolution, so causing it to be adopted as the decision, (e.g. *"Yeah"*, *"Good"* and *"Okay"*.

(1) A: Are we going to have a backup?
Or we do just–
B: But would a backup really be necessary?
A: I think maybe we could just go for the kinetic energy and be bold and innovative.
C: Yeah.
B: I think– yeah.
A: It could even be one of our selling points.
C: Yeah –*laugh*–.
D: Environmentally conscious or something.
A: Yeah.
B: Okay, fully kinetic energy.
D: Good.[2]

3.2 Experimental data for real-time decision detection

Originally, two individuals used the annotation scheme described above in order to annotate the manual transcripts of 9 and 10 meetings respectively. The annotators overlapped on two meetings, and their *kappa* inter-annotator agreement ranged from 0.63 to 0.73 for the four DDA classes. The highest agreement was obtained for class *RP*, and the lowest for class *A*. Although these kappa values are not extremely high, if we used a single, less homogeneous "decision-related" DA class like Hsueh and Moore (2007), then its kappa score

[1] This information was obtained through personal communication.

[2] This example was extracted from the AMI dialogue ES2015c and has been modified slightly for presentation purposes.

key	DDA class	description
I	*issue*	utterances introducing the issue or topic under discussion
R	*resolution*	utterances containing the resolution adopted as the decision
RP	*– proposal*	– utterances where the decision is originally proposed
RR	*– restatement*	– utterances where the decision is confirmed or restated
A	*agreement*	utterances explicitly signalling agreement with the decision

Table 1: Set of decision dialogue act (DDA) classes

would probably be significantly lower. The decision discussion annotations used by Hsueh and Moore (2007) are part of the AMI corpus, and are for the manual transcriptions. The reader can find a comparison between these annotations and our own manual transcript annotations in (Fernández et al., 2008).

After obtaining the new off-line and real-time ASR transcripts, we transferred the DDA annotations from the manual transcripts. In both sets of ASR transcripts, each meeting contains on average around 26 DAs tagged with one or more of the DDA sub-classes in Table 1. DDAs are thus very sparse, corresponding to only 5.3% of utterances in the real-time transcripts, and 6.0% in the off-line. In the real-time transcripts, *Issue* utterances make up less than 1.2% of the total number of utterances in a meeting, while *Resolution* utterances are around 1.6%: 1.2% are *RP* and less than 0.4% are *RR* on average. Almost half of DDA utterances (slightly over 2.6% of all utterances on average) are tagged as belonging to class *Agreement*. In the off-line transcripts, the percentages are fairly similar: 1.6% of utterances are *Issue* DDAs, 2.0% are *RP*, 0.5% are *RR*, and 2.4% are *A*.

We now move on to describe the *hierarchical classification* approach which we use to try to automatically detect decision sub-dialogues and their component DDAs.

4 Hierarchical Classification

Hierarchical classification is designed to exploit the fact that within decision discussions, our DDAs can be expected to co-occur in particular types of patterns. It involves two different types of classifier:

1. **Sub-classifier:** One independent binary *sub-classifier* per DDA class classifies each utterance.

2. **Super-classifier:** A sliding window shifts through the meeting one utterance at a time, and following each shift, a binary *super-classifier* determines whether the region of dialogue within the window is part of a decision discussion.

In our decision detectors, the sub-classifiers run in parallel in order to reduce processing time. For each utterance, the sub-classifiers use features which are derived from the properties of that utterance in context. On the other hand, the super-classifier's features are the hypothesized class labels and confidence scores for the utterances within the window. In various experiments, we have found that a suitable size for the window, is the average length of a decision discussion in our data in utterances. The super-classifier also "corrects" the sub-classifiers. This means that if a DA is classified as positive by a sub-classifier, but does not fall within a region classified as part of a decision discussion by the super-classifier, then the sub-classifier's hypothesis is changed to negative.

We now move on to consider how this basic approach to decision detection can be implemented in a real-time system.

5 Design considerations for our real-time system

A real-time decision detector should detect decisions as soon after they are made as possible. It is for this reason that we have set our real-time detector to automatically run at frequent and regular intervals during a meeting. An alternative would be to give the user (a meeting participant) responsibility for instructing the detector when to run. However, a user may sometimes leave substantial gaps between giving run commands. When this happens, the detector will have to process a large number of utterances in a single run, and so the user may wait some time before being presented with any results. In addition, giving the user responsibility for instructing the detector when to

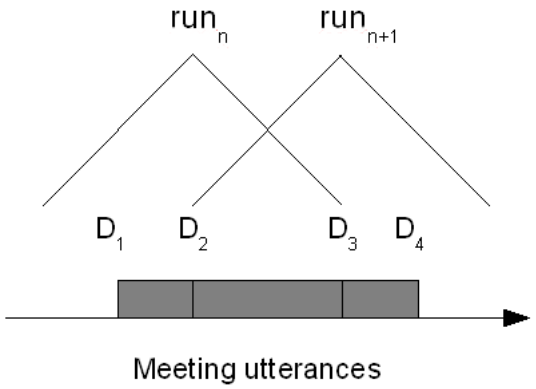


Figure 1: Decision discussion regions hypothesized by consecutive runs overlap (D_1 to D_3 and D_2 to D_4) and so are merged.

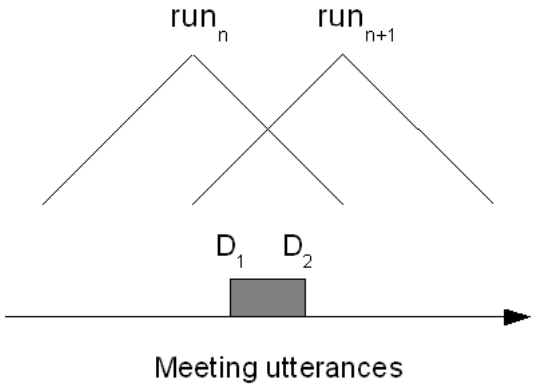


Figure 2: Consecutive runs hypothesize the same decision discussion region D_1 to D_2, and so one of the duplicates is discarded.

run means burdening the user with an extra task to perform during the meeting, and this goes against the general philosophy behind the system's development. The system is intended to be as unobtrusive as possible during the meeting, and to relieve users of tasks which distract their attention away from the current discussion (e.g. note-taking), not to create new tasks, however small.

Obviously, on the first occasion that the detector runs during a meeting, it can only process "new" (previously unprocessed) utterances, but on subsequent runs, it has the option to reprocess some number of "old" utterances (utterances which it has already processed in a previous run). Certainly, the detector should reprocess some of the most recent old utterances because it is possible that a decision discussion straddles these utterances and new utterances. However, the number of old utterances that are reprocessed should be limited. If the meeting has lasted a while already, then the processing of a large portion of the earlier old utterances is likely to be redundant — it will simply produce the same results for these utterances as the previous run.

The fact that the real-time detector processes recent old utterances means that consecutive runs can produce hypotheses for decision discussion regions which overlap, or which are duplicates. Figure 1 gives an example of the former. We deal with overlapping hypotheses by merging them into one, so forming a larger single decision discussion region. Figure 2 gives an example of duplicate hypotheses. Here, on run n, the detector hypothesizes decision discussion D_1 to D_2, and then on run $n+1$, since the bounds of this original hypothesis are now wholly contained within the region of old reprocessed utterances, the detector hypothesizes a duplicate. We deal with such cases by discarding the duplicate.

6 Experiments

We conducted various experiments related to real-time decision detection, our goal being to produce a system which:

- relative to alternative versions, detects decision discussions accurately,
- generates results for any portion of dialogue very soon after that portion of dialogue has ended.

The current version of our real-time detector is set to process the same number of old and new utterances on each run. Here, we refer to this value as i, and hence on each run the system processes a total of $2i$ utterances (i old and i new). Another of the system's characteristics is that runs take place every i utterances, meaning that as we decrease i, the system provides new results more frequently and is hence "more real-time". One of the things we investigate here then, is what to set i to in order to best satisfy the two design goals given above. Having found this value, we compare the hierarchical real-time detector's performance with alternative detectors, these being:

- an off-line detector applied to off-line ASR transcripts,
- a flat real-time detector,
- an off-line detector applied to the real-time ASR transcripts.

Lexical	unigrams after text normalization
Utterance	length in words, duration in word rate
Speaker	speaker ID & AMI speaker role
Context	features as above for utterances u +/- 1...u +/- 5

Table 2: Features for decision DA detection

Note that the off-line detectors use hierarchical classification, and that the flat real-time detector uses a single binary classifier which treats all DDAs as members of a single merged DDA class.

6.1 Classifiers and features

All classifiers (sub and super-classifiers) in all detectors are linear-kernel *Support Vector Machines (SVMs)*, produced using *SVMlight* (Joachims, 1999). For the sub-classifiers, we are obviously restricted to using features which can be computed in a very short period of time, and in the experiments here, we use *lexical*, *utterance* and *speaker* features. These are summarized in Table 2. An utterance's lexical features are the words in its transcription, its utterance features are its duration, number of words, and word rate (number of words divided by duration), and its speaker features are the speaker's role (see Section 3) and ID. We also use lexical features for the previous and where available, next utterances: the *I*, *RP* and *RR* sub-classifiers use the lexical features for the previous/next utterance and the *A* sub-classifier, those from the previous/next 5 utterances. These settings produced the best results in preliminary experiments. We do not use DA features because we lack an automatic DA tagger, nor do we use prosodic features because (Fernández et al., 2008) was unable to derive any value from them with SVMs.

6.2 Evaluation

We evaluate each of our decision detectors in 17-fold cross validations, where in each fold, the detector trains on 16 meetings and then tests on the remaining one. Evaluation can be made at three levels:

1. The sub-classifiers' detection of each of the DDA classes.

2. The sub-classifiers' detection of each of the DDA classes after correction by the super-classifier.

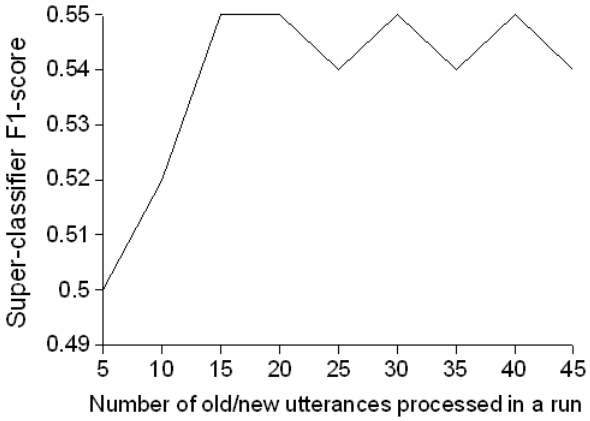


Figure 3: The relationship between the number of old/new utterances processed in a single run, and the super-classifier's F1-score. Here the sub-classifiers use only lexical features.

3. The super-classifier's detection of decision discussion regions.

For 1 and 2, we use the same lenient-match metric as (Fernández et al., 2008; Hsueh and Moore, 2007), which allows a margin of 20 seconds preceding and following a hypothesized DDA. Note that here we only give credit for hypotheses based on a 1-1 mapping with the gold-standard labels. For 3, we follow (Fernández et al., 2008; Purver et al., 2007) and use a windowed metric that divides the dialogue into 30-second windows and evaluates on a per window basis.

6.3 Results and analysis

Here, Section 6.3.1 will present results for different values of i, the number of old/new utterances processed in a single run. Section 6.3.2 then compares the performance of the real-time and off-line systems, (and also real-time systems which use hierarchical *vs.* flat classification), and Section 6.3.3 presents some feature analysis.

6.3.1 Varying the number of old/new utterances processed in a run

Figure 3 shows the relationship between i, the setting for the number of old/new utterances processed in a single run, and the super-classifier's F1-score. Here, the sub-classifiers are using only lexical features. We can see from the graph that as i increases to 15, the super-classifier's F1-score also increases, but thereafter, it plateaus. Hence 15 is apparently the value which best satisfies the two design goals given at the start of Section 6. It should also be noted that 15 is the mean length of a decision discussion in our data, and so per-

	sub-classifiers				super
	I	RP	RR	A	classifier
Re	.73	.73	.84	.71	.82
Pr	.08	.09	.03	.15	.40
F1	.15	.16	.06	.25	.54

Table 3: Results for the hierarchical real-time decision detector, using lexical, utterance and speaker features.

	sub-classifiers				super
	I	RP	RR	A	classifier
Re	.51	.51	.10	.63	.83
Pr	.12	.11	.04	.15	.41
F1	.19	.19	.05	.24	.55

Table 4: Results for the hierarchical off-line decision detector on off-line ASR transcripts, using lexical, utterance and speaker features.

haps this is a transferable finding. The mean duration of a run when $i = 15$ is approximately 4 seconds, while the mean duration of 15 utterances in our data-set is approximately 60 seconds, meaning that for the average case, the detector returns the results for the current run, long before it is due to make the next. Significant lee-way is perhaps necessary here, because the final version of the real-time detector will include a summarization component which extracts key phrases from *Issue/Resolution* utterances, and its processing can last some time, even for a single decision.

We should say then, that the system is not strictly real-time because in general, it detects decisions soon after they are made (for example within a minute), rather than immediately after. In the future we intend to modify the system so that it can run more frequently than once every i utterances. However it is important that runs do not occur too frequently — for example, if $i = 15$ and the system runs after every utterance, then the extra processing will cause it to gradually fall further and further behind the meeting.

6.3.2 Real-time *vs.* off-line results

Table 3 shows the results achieved by a hierarchical real-time decision detector whose run settings are as described above, and whose sub-classifiers[3] use lexical, utterance and speaker features. These results compare well with those of an equivalent off-line detector, which are shown in Table 4. The F1-scores for the real-time and off-line decision super-classifiers are $.54$ and $.55$ respectively, and the difference is not statistically significant. This may indicate that the hierarchical classification approach is fairly robust to increasing ASR Word Error Rates (WERs). Combining the output from each of the independent sub-classifiers might compensate somewhat for any decreases in their individual accuracy, as there was here for the *I* and *RP* sub-classifiers.

[3] In Tables 3 to 6, sub-classifier results are post-correction (see Section 6.2).

	sub-classifiers				super
	I	RP	RR	A	classifier
Re	.50	.51	.09	.63	.83
Pr	.11	.11	.03	.14	.41
F1	.19	.18	.05	.23	.55

Table 5: Results for the hierarchical off-line detector on real-time ASR transcripts, using lexical, utterance and speaker features.

	sub-classifiers				super
	I	RP	RR	A	classifier
Re	.67	.74	.84	.66	.85
Pr	.07	.08	.03	.14	.41
F1	.13	.15	.05	.24	.55

Table 6: Results for the hierarchical real-time decision detector, using lexical features only.

The hierarchical real-time detector's F1-score is also 10 points higher than a flat classifier ($.54$ *vs.* $.44$). Hence, while Fernández et al. (2008) demonstrated that the hierarchical classification approach could improve off-line decision detection, we have demonstrated here that it can also improve real-time decision detection.

Table 5 shows the results when an off-line detector is applied to real-time ASR transcripts. Here, the super-classifier obtains an F1-score of $.55$, one point higher than the real-time detector, but again, the difference is not statistically significant.

6.3.3 Feature analysis

We also investigated the contribution of the utterance and speaker features. Table 6 shows the results for the hierarchical real-time decision detector when its sub-classifiers use only lexical features. The sub-classifier F1-scores are all slightly lower than when utterance and speaker features are used (see Table 3), and the super-classifier

score is only 1 point different. None of these differences are statistically significant.

Since lexical features are important, we used *information gain* in order to investigate which words are predictive of each DDA type. Due to differences in the transcripts, the predictive words for the off-line and real-time systems are not the same, but we can find commonalities, and these commonalities make sense given the DDA definitions. Firstly in *Resolution* and particularly *Issue* DAs, some of the most predictive words could be used to define discussion topics, and so we might expect to find them in the meeting agenda. Examples are "energy", and "color". Predictive words for *Resolutions* also include semantically-related words which are key in defining the decision ("kinetic","green"). Additional predictive words for *RPs* are the personal pronouns "I" and "we", and the verbs, "think" and "like", and for *RRs*, words which we would associate with summing up ("consensus", "definitely", and "okay"). Unsurprisingly, for *Agreements*, "yeah" and "okay" both score very highly.

7 Conclusion

(Fernández et al., 2008) described an approach to decision detection in multi-party meetings and demonstrated how it could work relatively well in an off-line system. The approach has two defining characteristics. The first is its use of an annotation scheme which distinguishes between different utterance types based on the roles which they play in the decision-making process. The second is its use of hierarchical classification, whereby binary *sub-classifiers* detect instances of each of the decision DAs (DDAs), and then based on the sub-classifier hypotheses, a *super-classifier* determines which regions of dialogue are decision discussions.

In this paper then, we have taken the same basic approach to decision detection as Fernández et al. (2008), but changed the way in which it is implemented so that it can work effectively in real-time. Our implementation changes include running the detector at regular and frequent intervals during the meeting, and reprocessing recent utterances in case a decision discussion straddles these and brand new utterances. The fact that the detector reprocesses utterances means that on consecutive runs, overlapping and duplicate hypothesized decision discussions are possible. We have therefore added facilities to merge overlapping hypotheses and to remove duplicates.

In general, the resulting system is able to detect decisions soon after they are made (for example within a minute), rather than immediately after. It has performed well in testing, achieving an F1-score of .54, which is only one point lower than an equivalent off-line system, and in any case, the difference was not statistically significant. A flat real-time detector achieved .44.

In future work, we plan to extend the decision discussion annotation scheme and try to extract supporting arguments for decisions. We will also experiment with using sequential models in order to try to exploit any sequential ordering patterns in the occurrence of the DDAs.

Acknowledgements This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185/0004, and by the Department of the Navy Office of Naval Research (ONR) under Grants No. N00014-05-1-0187 and N00014-09-1-0106. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or ONR. We are grateful to the three anonymous EMNLP reviewers for their helpful comments and suggestions, and to our partners at SRI International who provided us with off-line and real-time transcripts for our meeting data.

References

Satanjeev Banerjee, Carolyn Rosé, and Alex Rudnicky. 2005. The necessity of a meeting recording and playback system, and the benefit of topic-level annotations to meeting browsing. In *Proceedings of the 10th International Conference on Human-Computer Interaction*.

Susanne Burger, Victoria MacLaren, and Hua Yu. 2002. The ISL Meeting Corpus: The impact of meeting type on speech style. In *Proceedings of the 7th International Conference on Spoken Language Processing (INTERSPEECH - ICSLP)*, Denver, Colorado.

Raquel Fernández, Matthew Frampton, Patrick Ehlen, Matthew Purver, and Stanley Peters. 2008. Modelling and detecting decisions in multi-party dialogue. In *Proc. of the 9th SIGdial Workshop on Discourse and Dialogue*.

Michel Galley, Kathleen McKeown, Julia Hirschberg, and Elizabeth Shriberg. 2004. Identifying agree-

ment and disagreement in conversational speech: Use of Bayesian networks to model pragmatic dependencies. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Daniel Gatica-Perez, Ian McCowan, Dong Zhang, and Samy Bengio. 2005. Detecting group interest level in meetings. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Dustin Hillard, Mari Ostendorf, and Elisabeth Shriberg. 2003. Detection of agreement vs. disagreement in meetings: Training with unlabeled data. In *Companion Volume of the Proceedings of HLT-NAACL 2003 - Short Papers*, Edmonton, Alberta, May.

Pey-Yun Hsueh and Johanna Moore. 2007. Automatic decision detection in meeting speech. In *Proceedings of MLMI 2007*, Lecture Notes in Computer Science. Springer-Verlag.

Adam Janin, Jeremy Ang, Sonali Bhagat, Rajdip Dhillon, Jane Edwards, Javier Marcías-Guarasa, Nelson Morgan, Barbara Peskin, Elizabeth Shriberg, Andreas Stolcke, Chuck Wooters, and Britta Wrede. 2004. The ICSI meeting project: Resources and research. In *Proceedings of the 2004 ICASSP NIST Meeting Recognition Workshop*.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press.

Agnes Lisowska, Andrei Popescu-Belis, and Susan Armstrong. 2004. User query analysis for the specification and evaluation of a dialogue processing and retrieval system. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*.

Iain McCowan, Jean Carletta, W. Kraaij, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, W. Post, D. Reidsma, and P. Wellner. 2005. The AMI Meeting Corpus. In *Proceedings of Measuring Behavior, the 5th International Conference on Methods and Techniques in Behavioral Research*, Wageningen, Netherlands.

Matthew Purver, John Dowding, John Niekrasz, Patrick Ehlen, Sharareh Noorbaloochi, and Stanley Peters. 2007. Detecting and summarizing action items in multi-party dialogue. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*, Antwerp, Belgium.

Andreas Stolcke, Xavier Anguera, Kofi Boakye, Özgür Çetin, Adam Janin, Matthew Magimai-Doss, Chuck Wooters, and Jing Zheng. 2008. The ICSI-SRI Spring 2007 meeting and lecture recognition system. In *Proc. of CLEAR 2007 and RT2007*.

Daan Verbree, Rutger Rienks, and Dirk Heylen. 2006. First steps towards the automatic construction of argument-diagrams from real discussions. In *Proceedings of the 1st International Conference on Computational Models of Argument*, volume 144, pages 183–194. IOS press.

A. Waibel, T. Schultz, M. Bett, M. Denecke, R. Malkin, I. Rogina, R. Stiefelhagen, and J. Yang. 2003. SMaRT: The smart meeting room task at ISL. In *ICASSP*.

Steve Whittaker, Rachel Laban, and Simon Tucker. 2006. Analysing meeting records: An ethnographic study and technological implications. In S. Renals and S. Bengio, editors, *Machine Learning for Multimodal Interaction: Second International Workshop, MLMI 2005, Revised Selected Papers*, volume 3869 of *Lecture Notes in Computer Science*, pages 101–113. Springer.

Britta Wrede and Elizabeth Shriberg. 2003. Spotting "hot spots" in meetings: Human judgements and prosodic cues. In *Proceedings of the 9th European Conference on Speech Communication and Technology*, Geneva, Switzerland.

On the Role of Lexical Features in Sequence Labeling

Yoav Goldberg* and **Michael Elhadad**
Ben Gurion University of the Negev
Department of Computer Science
POB 653 Be'er Sheva, 84105, Israel
{yoavg|elhadad}@cs.bgu.ac.il

Abstract

We use the technique of SVM anchoring to demonstrate that lexical features extracted from a training corpus are not necessary to obtain state of the art results on tasks such as Named Entity Recognition and Chunking. While standard models require as many as 100K distinct features, we derive models with as little as 1K features that perform as well or better on different domains. These robust reduced models indicate that the way rare lexical features contribute to classification in NLP is not fully understood. Contrastive error analysis (with and without lexical features) indicates that lexical features do contribute to resolving some semantic and complex syntactic ambiguities – but we find this contribution does not generalize outside the training corpus. As a general strategy, we believe lexical features should not be directly derived from a training corpus but instead, carefully inferred and selected from other sources.

1 Introduction

Common NLP tasks, such as Named Entity Recognition and Chunking, involve the identification of spans of words belonging to the same phrase. These tasks are traditionally reduced to a tagging task, in which each word is to be classified as either **B**eginning a span, **I**nside a span, or **O**utside of a span. The decision is based on the word to be classified and its neighbors. Features supporting the classification usually include the word forms themselves and properties derived from the word forms, such as prefixes, suffixes, capitalization information, and parts-of-speech. While early approaches to the NP-chunking task (Cardie and Pierce, 1998) relied on part-of-speech information alone, it is widely accepted that lexical information (word forms) is crucial for building accurate systems for these tasks. Indeed, all the better-performing systems in the CoNLL shared tasks competitions for Chunking (Sang and Buchholz, 2000) and Named Entity Recognition (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) make extensive use of such lexical information.

Is this belief justified? In this paper, we show that the influence of lexical features on such sequence labeling tasks is more complex than is generally assumed. We find that exact word forms aren't necessary for accurate classification. This observation is important because relying on the exact word forms that appear in a training corpus leads to over-fitting, as well as to larger models.

In this work, we focus on learning with Support Vector Machines (SVMs) (Vapnik, 1995). SVM classifiers can handle very large feature spaces, and produce state-of-the-art results for NLP applications (see *e.g.* (Kudo and Matsumoto, 2000; Nivre et al., 2006)). Alas, when trained on pruned feature sets, in which rare lexical items are removed, SVM models suffer a loss in classification accuracy. It would seem that rare lexical items are indeed crucial for SVM classification performance. However, in Goldberg and Elhadad (2007), we suggested that the SVM learner is using the rare lexical features for singling out hard cases rather than for learning meaningful generalizations. We provide further evidence to support this claim in this paper.

*Supported by the Lynn and William Frankel Center for Computer Sciences, Ben Gurion University

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1142–1151,
Singapore, 6-7 August 2009.

We show that by using a variant of SVM – *Anchored SVM Learning* (Goldberg and Elhadad, 2007) with a polynomial kernel, one can learn accurate models for English NP-chunking (Marcus and Ramshaw, 1995), base-phrase chunking (CoNLL 2000), and Dutch Named Entity Recognition (CoNLL 2002), on a heavily pruned feature space. Our models make use of only a fraction of the lexical features available in the training set (less than 1%), and yet provide highly-competitive accuracies.

For the Chunking and NP-Chunking tasks, the most heavily pruned experiments, in which we consider only features appearing at least 100 times in the training corpus, do show a small but significant drop in accuracy on the testing corpus compared to the non-pruned models exposed to all available features in the training data. We provide detailed error analysis of a development set in Section 6, revealing the causes for these differences. We suggest one additional binary feature in order to account for some of the performance gap. Moreover, we show that the differences in accuracy vanish when the lexicalized and unlexicalized models are tested on text from slightly different sources than the training corpus (Section 7).

This goes to show that with an appropriate learning method, orthographic and structural (in the form of POS tag sequences) information is sufficient for achieving state-of-the-art performance on these kind of sequence labeling tasks. This does not mean semantic information is not needed for these tasks. It does mean that current models capture only a tiny amount of such semantic information through rare lexical features, and in a manner that does not generalize well.

We believe this data motivates a different strategy to incorporate lexical features into classification models: instead of collecting the raw lexical forms appearing in a training corpus, we should attempt to actively construct a feature space including lexical features derived from external sources. The feature representation of (Collobert and Weston, 2008) could be a step in that direction. We also believe that hard cases for sequence labeling (POS ambiguity, coordination, long syntactic constructs) could be directly approached with specialized classifiers.

1.1 Related Work

This work complements a similar line of results from the parsing literature. While it was initially believed that lexicalization of PCFG parsers (Collins, 1997; Charniak, 2000) is crucial for obtaining good parsing results, Gildea (2001) demonstrated that the lexicalized Model-1 parser of Collins (1997) does not benefit from bilexical information when tested on a new text domain, and only marginally benefits from such information when tested on the same text domain as the training corpora. This was followed by (Bikel, 2004) who showed that bilexical-information is used in only 1.49% of the decisions in Collins' Model-2 parser, and that removing this information results in "an exceedingly small drop in performance". However, uni-lexical information was still considered crucial. Klein and Manning (2003) bridged the gap between lexicalized and unlexicalized parsing performance, providing a competitive unlexicalized parsing model, relying on lexical information for only a few closed-class lexical items. This was recently followed by (Matsuzaki et al., 2005; Petrov et al., 2006) who introduce state-of-the-art nearly unlexicalized PCFG parsers.

Similarly for discriminative dependency parsing, state-of-the-art parsers (McDonald, 2006; Nivre et al., 2006) are highly lexicalized. However, the model analysis in (McDonald, 2006) reveals that bilexical features hardly contribute to the performance of a discriminative MST-based dependency parser, while Kawahara and Uchimoto (2007) demonstrate that minimally-lexicalized shift-reduce based dependency parsers can produce near state-of-the-art accuracy.

In this work, we address the same question of determining the impact of lexical features on a different family of tasks: sequence labeling, as illustrated by named entity recognition and chunking. As discussed above, all state-of-the-art published methods rely on lexical features for such tasks (Zhang et al., 2001; Sha and Pereira, 2003; Finkel et al., 2005; Ratinov and Roth, 2009). Sequence labeling includes both a structural aspect (bracketing the chunks) and a tagging aspect (classifying the chunks). While we expect the structural aspect can benefit from techniques similar to those used in the parsing literature, it is unclear whether the tagging component could perform well without detailed lexical information. We demonstrate in this work that, indeed, lexical features are not necessary to obtain competitive performance. Our approach consists in performing a detailed analysis

of the role played by rare lexical features in SVM models. We distinguish the information brought to the model by such features from the role they play in a specific learning method.

2 Learning with Less Features

We adopt the common feature representation in which each data-point is represented as a sparse D dimensional binary-valued vector f. Each of the D possible features f_i is an indicator function. The indicator functions look at properties of the current or neighbouring words. An example of such function f_i is `1 iff the previous word-form is DOG, 0 otherwise`. The lexical (word-form) features result in extremely high-dimensional (yet very sparse) feature vectors – each word-form in the vocabulary of the training set correspond to (at-least) one indicator function.

Due to the Zipfian distribution of language data, many of the lexical features are very rare, and appear only a couple times in the training set. Ideally, we would like our classifiers to learn only from robust features: consider only features that appear at least k times in the training data (rare-feature pruning). These features are more likely to appear in unseen test data, and thus such features can support more robust generalization.

However, we find empirically that performing such feature pruning *prior* to learning SVM models hurts the performance of the learned models. Our intuition is that this sensitivity to rare lexical features is not explained by the richness of information such rare features bring to the model. Instead, we believe that rare lexical features help the classifier because they make the data *artificially* more separable. To demonstrate this claim, we experiment with anchored SVM, which introduces artificial mechanical anchors into the model to achieve separability, and make rare lexical features unnecessary.

3 Learning Method

SVM are discriminative, max-margin, linear classifiers (Vapnik, 1995), which can be kernelized. For the formulation of SVMs in the context of NLP applications, see (Kudo and Matsumoto, 2001). SVMs with a polynomial kernel of degree 2 were shown to provide state-of-the-art performance in many NLP application, see for example (Kudo and Matsumoto, 2000; Nivre et al., 2006; Isozaki and Kazawa, 2002; Goldberg et al., 2006).

SVMs cope with inseparable data by introducing a *soft-margin* – allowing some of the training instances to be classified incorrectly subject to a penalty, controlled by a parameter C.

Anchored SVM As we show in Section 5, the soft-margin heuristic performs sub-optimally for NLP tasks when the data is inseparable. We use instead the *Anchored Learning* heuristic, introduced in (Goldberg and Elhadad, 2007). The idea behind anchored learning is that some training instances are inherently ambiguous. This ambiguity stems from ambiguity in language structure, which cannot be resolved with a given feature representation. When a data-point cannot be classified, it might be due to missing information, which is not available in the data representation. Instead of allowing ambiguous items to be misclassified during training, we make the training data artificially separable. This is achieved by adding a unique feature to each training example (an *anchor*). These anchor features cause each data-point to be slightly more similar to itself than to any other data point. At test time, we remove anchor features.
In terms of kernel-based learning, anchored learning can be achieved by redefining the dot product between two vectors to take into account the identity of the vectors: $x_i \cdot_{anc} x_j = x_i \cdot x_j + \delta_{ij}$.

The classifier learned over the anchored data takes into account the fine interactions between the various inseparable data points. In our experiments, SVM models over anchored data have many more support vectors than soft-margin SVM models. However, the anchored models generalize much better when less features are available.

Relation to L2 SVM The classic soft-margin SVM formulation uses L1-penalty for misclassified instances. Specifically, the objective of the learner is to minimize $\frac{1}{2}||w||^2 + C\sum_i \xi_i$ subject to some margin constraints, where w is a weight vector to be learned and ξ_i is the misclassification error for instance i. This is equivalent to maximizing the dual problem:
$\sum_{i=1}^{M} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$

Another variant is L2-penalty SVM (Koshiba and Abe, 2003), in which there is a quadratic penalty for misclassified instances.
Here, the learning objective is to minimize: $\frac{1}{2}||w||^2 + \frac{1}{2}C\sum_i \xi_i^2$ or alternatively maximize the dual: $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (K(x_i, x_j) + \frac{\delta_{ij}}{C})$.

Interestingly, for the linear kernel, SVM-

anchoring reduces to L2-SVM with C=1. However, for the case of non-linear kernels, anchored and L2-SVM produce different results, as the anchoring is applied prior to the kernel expansion. Specifically for the case of the second-degree polynomial kernel, L2-SVM aims to maximize:

$\sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j((x_i \cdot x_j + 1)^2 + \frac{\delta_{ij}}{C})$,

while the anchored-SVM variant would maximizes: $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j(x_i \cdot x_j + \delta_{ij} + 1)^2$.

In our experiments, as discussed in Section 5.4, we find that anchored-SVM and soft-margin SVM with tuned C value both reach good results when we reduce the amount of lexical features. Anchored-SVM, however, does not require fine-tuning of the error-parameter C since it insures separability. As a result, we learn anchored-SVM models quickly (few hours) as opposed to several days per model for C-tuned soft-margin SVM. Anchored-SVMs also provide an easy explanation of the role of features in terms of separability. Therefore, we use anchored-SVMs in our experiments as the learning method, but we expect that other learning methods are capable of learning with the same reduced feature sets.

4 Experiment Setup

How important are the rare lexical features for learning accurate NLP models? To investigate this question, we experiment with 3 different NLP sequence-labeling tasks. For each task, we train a sequence of polynomial kernel (d=2) SVM classifiers, using both soft-margin (C=1) and anchored SVM. Each classifier is trained on a pruned feature set, in which only features appearing at least k times in the training data are kept. We vary the pruning parameter k. Pruning is performed over all the features in the model, but lexical features are most affected by it.

For all the models, we use the B-I-O representation, and perform multiclass classification using pairwise-voting. For our features, we consider properties of tokens in a 5-token window centered around the token to be classified, as well as the two previous classifier predictions. Results are reported as F-measure over labeled identified spans.

Polynomial vs. Linear models The polynomial kernel of degree 2 allows us to efficiently and implicitly include in our models all feature pairs. Syntactic structure information as captured by pairs of POS-tags and Word-POS pairs is certainly important for such syntactic tasks as Chunking and NER, as demonstrated by the many systems described in (Sang and Buchholz, 2000; Tjong Kim Sang, 2002). By using the polynomial kernel, we can easily make use of this information without intensive feature-tuning for the most successful feature pairs.

L1-SVM, L2-SVM and the choice of the C parameter Throughout our experiments, we use the "standard" variant of SVM, L1-penalty soft margin SVM, as implemented by the TinySVM[1] software package, with the default C value of 1. This setting is shown to produce good results for sequence labeling tasks in previous work (Kudo and Matsumoto, 2000), and is what most end-users of SVM classifiers are likely to use. As we show in Sect.5.4, fine-tuning the C parameter reaches better accuracy than L1-SVM with C=1. However, as this fine-tuning is computationally expensive, we first report the comparison L1-SVM/C=1 vs. anchored-SVM, which consistently reached the best results, and was the quickest to train.

Feature Pruning vs. Feature Selection Our aim in this set of experiments is not to find the optimal set of lexical features, but rather to demonstrate that most lexical items are not needed for accurate classification in sequence labeling tasks. To this end, we perform very crude frequency based feature pruning. We believe better motivated feature selection technique taking into account linguistic (*e.g. prune only open-class words*) or statistic information could result in slightly more accurate models with even fewer lexical items.

5 Experiments and Results

5.1 Named Entity Recognition (NER)

We use the Dutch data set from the CoNLL 2002 shared task (Tjong Kim Sang, 2002). The aim is to identify named entities (persons, locations, organizations and miscellaneous) in text. The task has two stages: identification of the entities, and classification of the identified entities into their corresponding types. We focus here on the identification task.

Features: We use the following properties for each of the relevant tokens: word-form, POS, ORT, prefix1, prefix2, prefix3, suffix1, suffix2, suffix3. The ORT feature can take one of the following values: {number, contains-digit, contains-hyphen, capitalized, all-capitalized, URL, punctuation, regular}.

[1] http://chasen.org/~taku/software/TinySVM/

PRUNING	#FEATURES	SOFT-MARGIN	ANCHORED
0	186,421	90.92	90.78
100	5,804	90.73	90.75
1000	1,207	88.56	90.10
1500	821	85.92	89.29

Table 1: Named Entity Identification results (F-score) on dev set, with various pruning thresholds.

Results are presented in Table 1. Without feature pruning, we achieve an F-score of 90.9. This dataset proved to be quite resilient to feature pruning. Pruning features appearing less than 100 times results in just a slight decrease in F-score. Extremely aggressive pruning, keeping only features appearing more than 1,000 or 1,500 times in the training data, results in a big drop in F-score for the soft-margin SVM (from about 91 to 86). Much less so for the Anchored-SVM. Using Anchored SVM we achieve an F-score of 90.1 after pruning with $k = 1,000$. This model has 1207 active features, and 27 unique active lexical forms.

5.2 NP Chunking

The goal of this task (Marcus and Ramshaw, 1995) is the identification of non-recursive NPs. We use the data from the CoNLL 2000 shared task: NP chunks are extracted from Sections 15-18 (train) and 20 (test) of the Penn WSJ corpus. POS tagged are automatically assigned by the Brill Tagger.

Features: We consider the POS and word-form of each token.

PRUNING	#FEATURES	SOFT-MARGIN	ANCHORED
0	92,805	94.12	94.08
1	46,527	93.78	94.09
2	32,583	93.58	94.00
5	18,092	93.42	94.01
10	10,812	93.00	93.98
20	5,952	92.48	93.92
50	2,436	92.33	93.96
100	1,168	91.94	93.83

Table 2: NP-Chunking results (F-score), with various pruning thresholds.

Results are presented in Table 2. Without feature pruning ($k = 0$), the soft-margin SVM performs slightly better than the Anchored-SVM. Either of the results are state-of-the-art for this task. However, even modest pruning ($k = 2$) hurts the soft-margin model significantly. Not so for the anchored-SVM. Even with relatively aggressive pruning ($k = 100$), the anchored model still achieves an impressive F-score of 93.83. Remarkably, in that last model, there are only 1,168 active features, and only 209 unique active lexical forms.

5.3 Chunking

The goal of the Chunking task (Sang and Buchholz, 2000) is the identification of an assortment of linguistic base-phrases. We use the data from the CoNLL 2000 shared task.

Features: We perform two experiments. In the first experiment, we consider the POS and word-form of each token. In this setting, feature pruning resulted in a bigger loss in performance than in the two previous tasks. Preliminary error analysis revealed that many errors are due to tagger errors, especially of the present participle forms. This led us to the second experiment, in which we added as features the 2- and 3- letter suffixes for the word to be classified (but not for the surrounding words).

Results are presented in Tables 3 and 4. In the first experiment (POS + Word), the non-pruned soft-margin model is the same system as the top-performing system in the original shared task, and yields state-of-the-art results. Unlike the NP-chunking case, here feature pruning has a relatively large impact on the results even for the anchored models. However, the anchored models are still far more robust than the soft-margin ones. With $k = 100$ pruning, the soft-margin model suffers a drop of 2.5 F points, while the anchored model suffers a drop of only 0.84 F points. Even after this drop, the anchored $k = 100$ model still performs above the top-third system in the CoNLL 2000 shared task. This anchored $k = 100$ model has 1,180 active features, and only 209 unique active lexical features.

The second experiment (POS + word-form + suffixes for main word) adds crude morphological information to the learner, helping it to avoid common tagger mistakes. This additional information is helpful: pruning with $k = 100$ leads to an accurate anchored model (93.12 F) with only 209 unique lexical items. Note that with the addition of the suffix features, the pruned model $k = 20$ beats the purely lexical model (no suffix features) with no pruning (93.51 vs. 93.44) with 10 times less features. When we combine suffixes and all lexical forms, we still see a slight advantage to the lexical model (93.73 vs. 93.12 with pruning at $k = 100$).

Even less lexicalization How robust are the suffixes? We performed a third experiment, in which

PRUNING	#FEATURES	SOFT-MARGIN	ANCHORED
0	92,837	93.44	93.40
1	46,557	93.20	93.32
2	32,614	93.10	93.31
5	18,126	92.89	93.29
10	10,834	92.73	93.23
20	5,975	92.18	93.16
50	2,463	91.80	92.89
100	1,180	90.94	92.56

Table 3: Chunking results (F), with various pruning thresholds. Experiment 1. Features: POS, Word.

PRUNING	#FEATURES	SOFT-MARGIN	ANCHORED
0	104,304	93.73	93.69
1	72,228	93.56	93.68
2	57,578	93.50	93.64
5	37,210	93.35	93.62
10	23,968	93.26	93.56
20	14,060	92.84	93.51
50	6,326	92.28	93.37
100	3,340	91.83	93.12

Table 4: Chunking results (F), with various pruning thresholds. Experiment 2. Features: POS, Word, {Suff2, Suff3} of main Word.

we replaced any explicit word-forms by 2- and 3-letter suffixes. This gives us the complete word form of many function words, and a reasonable amount of morphological marking. Results are presented in Table 5. Surprisingly, this information proves to be quite robust. Without feature pruning, both the anchored and soft-margin model achieve near state-of-the-art performance of 93.25F. Pruning with $k = 100$ hurts the result of the soft-margin model, but the anchored model remains robust with an F-score of 93.18. This last model has 2,563 active features. With further pruning ($k = 250$), the result of the anchored model drops to 92.87F (still 3rd place in the CoNLL shared task), with only 1,508 active features in the model.

5.4 Fine-tuned soft-margin SVMs

For the sake of completeness, and to serve as a better comparison to the soft-margin SVM, we report results of some experiments with both L1 and L2 SVMs, with tuned C values. NP-chunking performance with tuned C values and various pruning thresholds is presented in Table 6.

For these results, the C parameter was tuned on a development set using Brent's 1-dimension minimization method (Brent, 1973). While taking about 40 hours of computation to fit, the final results catch up with those of the anchored-SVM but still remain slightly lower. This further highlights our main point: accurate models can be achieved also with mostly unlexicalized models, and the lexical features do not contribute substantial semantic information, but rather affect the separability of the data. This is nicely demonstrated by SVM-anchoring, in which lexical information is practically replaced by artificial semantically void indexes, but similar performance can also be achieved by fine-tuning other learning parameters.

PRUNING	#FEATURES	SOFT-MARGIN	ANCHORED
0	19,910	93.25	93.23
100	2,563	92.87	93.18
250	1,508	92.40	92.87

Table 5: Chunking results (F), with various pruning thresholds. Experiment 3. Features: POS , Suff2, Suff3 .

K	L1 (C)	L2 (C)	ANCHORED
0	94.12 (1.0001)	94.09 (2.6128)	94.08
50	93.79 (0.0524)	93.71 (0.0082)	93.96
100	93.72 (0.0567)	93.59 (0.0072)	93.83

Table 6: NP-Chunking results (F), with various pruning thresholds K, for L1 and L2 SVMs with tuned C values

6 Error Analysis

Our experiments so far indicate that very aggressive feature pruning hurts performance slightly (by about 0.5F point). The feature-pruned models are still accurate, indicating that lexical features contribute little to the classification accuracy. We now investigate the differences between the lexicalized and pruned models, in order to characterize the kind of information that is available to the lexicalized models but missing from the pruned ones. In the next section, we also verify that pruned-models are more stable than the fully lexicalized ones when tested over different text genres and domains.

We focus our analysis on the chunking task, which is a superset of the NP-chunking task. We compare the fully lexicalized soft-margin SVM model with the POS+suffix2+suffix3 anchored-SVM model with $k = 100$ pruning. We analyze the models' respective performance on section 05 of the WSJ corpus. This dataset is different than the official test set. It is, however, part of the same annotated corpus as both the training and test sets. On this dataset, the fully lexicalized SVM model

achieves an F-score of 93.24, vs. 92.59 for the suffix-based pruned anchored-SVM model. (The pruned anchored-SVM model ($k = 100$) from experiment 2, achieve a slightly higher F-score of 92.84)

We investigate only those chunks which are identified correctly by one model but not by the other. Overall, there are 440 chunks (363 unique) which are identified correctly only by the lexicalized model, and 258 chunks (232 unique) only by the pruned model.

Where the pruned model is always wrong

Some errors are unique to the pruned model.

Over 45 of the cases that are identified correctly only in the lexicalized model (more than 10%) are due to the words "including" (18 cases) and "If" (9 cases), as well as other -ing forms such as "following", "according", "rising" and "suspecting."

The word "including" appears 80 times in the training data, always tagged as VBG and functioning as a PP chunk, which is an odd chunk for VBGs. The lexicalized model easily picked up on this behaviour, while the pruned model couldn't. Similarly, the word "following/VBG" appears 32 times, 20 of which as PP, and the word "according/VBG" 53 times, all of them as PP. The pruned model could not distinguish those from the rest of the VBGs and tagged them as VPs. What seems to happen in these cases, is that certain verbal forms participate in idiomatic constructions and behave syntactically as prepositions. The POS tagger does not pick this ambiguity in function and contributes only the most likely tag for the words (VBG). Lexical models learn that certain VBGs are "becoming" prepositions in the observed dataset. These words do not appear as specific features in the pruned models, and hence these usage shifts are often misclassified. Interestingly, the pruned model did learn that verbal forms can sometimes be PPs: it made use of that information by mis-identifying 11 verbal VBGs and 6 verbal VBNs as PPs.

The word "If/IN", unlike most prepositions, it always starts an SBAR rather than a PP chunk in the corpus. The pruned model learned this behaviour correctly for the lower-cased "if/IN", but missed the upper-cased version appearing in 79 sentence initial locations in the corpus.

These cases are caused by a mismatch between the POS tag and the syntactic function observed in the chunked dataset.

Additional cases include the adverbs (Already, Nearby, Soon, Maybe, Perhaps, once, Then): they are sometimes not chunked as ADVP but are left outside of any chunk. Some one-word ADJP chunks being chunked as NPs (short, general, sure, worse, ...) (6 cases) and some are chunked as ADVPs (hard, British-born, ...) (4 cases).

There are 10 cases where the pruned model splits an NP into ADVP and NP, such as: [later] [this week], [roughly][18 more U.S. stores]. In addition, the pruned model failed to learn the construction "typical of", resulting in 2 NP chunks such as: [The more intuitive approach typical].

Some mistakes of the pruned model seem like mistakes/pecularities of the annotated corpus, which the lexicalized model found a way to work around. Consider the following gold-standard cases from the annotated corpus:

- [VP seems] [ADVP rarely] [VP to cut]
- [ADVP just] [PP after]
- [VP is] [NP anything] [O but] [VP fixing]
- [ADJP as high] [PP as] [NP 8.3 %]
- [ADJP less] [PP than] [ADJP rosy]
- [NP 40 %] [PP to] [NP 45 %]

Which were each identified as a single chunk by the pruned model. It can be argued these are mistakes in the tagged dataset.

Where the lexical model is sometimes better

Both models fail on conjunctions, but the lexicalized model do slightly better. Conjunction error types come in two main varieties, either chunking [x][and][y] instead of [x and y] (pruned: 21 cases, lex: 14 cases) or chunking [x and y] instead of [x][and][y] (pruned: 26 cases, lex: 24 cases).

Joining VP and NP into an NP, due to a verb/adj ambiguity. For example chunking [NP fired six executives] instead of [VP fired] [NP six executives], or [NP keeping viewers] instead of [VP keeping] [NP viewers]. 12 such cases are resolved correctly only by the lexicalized model, and 5 only by the pruned one.

SBAR/PP confusion for words such as: "as","after","with","since" (both ways). 13 cases for the pruned model, 6 for lexicalized one.

Where both model are similar

Merging back-to-back NPs: Both models tend to erroneously join back-to-back NPs to a single NP, *e.g.* : [NP Westinghouse this year], or [NP themselves fashion enterprises]. No model is better than the other on these cases, each model failed

on 16 cases the other model succeeded on.

Joining NP and VP into an NP due to Verb/Noun ambiguity and tagger mistakes:

- [NP the weekend] [VP making] → [NP the weekend making]
- [NP the competition] [VP can] → [NP the competition can]

(lexicalized: 6 errors, pruned: 8 errors)

And splitting some NPs to VP+NP due to the same reasons:

- [VP operating] [NP profit]
- [VP improved] [NP average yield]

(lexicalized: 5 errors, pruned: 7 errors)

The word "that" is confused between SBAR and NP (5 mistakes for each model)

Erroneously splitting range NPs, *e.g.* :

- [about $115][to][$125] (2 cases for each model).

Where the pruned model is better

There are some cases where the pruned models is doing better than the lexicalized one:

VP wrongly split into VP and ADJP:

- [remains] [banned]

4 mistakes for lexicalized, 1 for pruned

VP wrongly split into VP and VP:

- [were scheduled] [to meet]
- [used] [to complain]

3 mistakes for lexicalized, 1 for pruned

VP wrongly split into ADVP and VP:

- [largly][reflecting]
- [selectively][leaking]

6 mistakes for lexicalized, 1 for pruned

PP and SBAR confusion:

- of, with, As, after

9 mistakes for lex, 5 for pruned

VP chunked as NP due to tagger mistake:

- [NP ruling], [NP drives], [NP cuts]

6 mistakes for lex, 2 for pruned

"that" tagged as NP instead of SBAR:

2 mistakes for lex, 0 for pruned

To conclude

Both the pruned and the fully lexicalized models have problems dealing with non-local phenomena such as coordination and relative clauses, as well as verb/adjective ambiguities and VBG/Noun ambiguities. They also perform poorly on embeded syntactic constructions (such as an NP containing an ADJP), and on identification of back-to-back NPs, which often requires semantic knowledge.

Both models suffer from tagging mistakes of the underlying tagger and systematic ambiguity between the morphological tag assigned by the tagger and the syntactic tag in which the word operates (e.g., "including" used as a preposition).

The main advantage of the fully lexcialized model is in dealing with:

- Some coordinated constructions.
- Some cases of verb/adjective ambiguities.
- Specific function words not seen much in training.
- Idiomatic usages of some VBG/VBN forms functioning as prepositions.

The first two items are semantic in nature, and hint that lexical features do capture some semantic information. While this might be true on the specific corpus, we believe that such corpus-derived semantic knowledge is very restricted, is not generalizable, and will not transfer well to other corpora, even on the same genre. We provide evidence for this claim in Section 7.

The last two items are syntactic. We address them by introducing a slightly modified feature model.

6.1 Another chunking Experiment

Based on the observations from the error analysis, we performed another pruned-chunking experiment, with the following features:

- Word and POS for a -2,+2 window around the current token, and 2-and-3-letter suffixes of the token to be classified (same as Experiment 2 in Section 5.2 above).
- Features of words appearing as a preposition (IN) anywhere in the training set are not pruned (this result in a model with 310 unique lexical items after $k = 100$ pruning).
- *An additional binary feature* indicating for each token whether it can function as a PP. The list of possible-PP forms is generated by considering all tokens seen inside a PP in the training corpus. It can be easily extended if additional lexicographic resources are available, without retraining the model.

This last proposed feature incorporates important lexical knowledge without relying on features for specific lexical forms, and is more generalizable. The accuracy of this new model on the development and test set with various pruning thresholds is presented in Table 7.

The addition of the CanBePrep feature improves the fully-lexicalized model accuracy on the development set (93.24 to 93.68), and does not affect fully lexicalized result on the test set (93.71

CORPUS	SOURCE	CONTENT	#TOKENS
WSJ	4 articles from wsj.com business	Magazine, business	2,671
Jaguar	Wikipedia page on Jaguar	Well edited text, animals	5,396
FreeWill	Wikipedia page on Free Will	Well edited text, philosophy	9,428
LJ-Life	4 LiveJournal posts	Noisy teenage writing, life	870

Table 8: Corpus Variation Text Sources

PRUNING	#FEATURES	SOFT-MARGIN	ANCHORED
Dev Set			
0	92,989	93.71	–
100	4,066	–	93.22
Test Set			
0	92,989	93.68	–
100	4,066	–	93.26

Table 7: Chunking results (F), with various pruning thresholds. Experiment 4. Features: POS, Word , Suff2, Suff3 for main word, CanBePrep .

vs. 93.73). The pruned model performance improves in both cases, more so on the development set (93.12 to 93.22 on the test set, 92.84 to 93.26 on the development set). The new model helps bridging the gap between the fully lexicalized and the pruned model, yet we still observe a lead of 0.4F for the fully lexicalized model. We now turn to explore how meaningful this difference is in real-world situation in which one does not operate on the Penn-WSJ corpus.

7 Corpus Variation and Model Performance

When tested on the exact same resource as the models are trained on, the fully lexicalized model still has a slight edge over the pruned ones. How well does this lexical knowledge transfer to different text genres? We compare the models' performance on text from various genres, ranging from very similar to the training material (recent articles from the WSJ Business section) to a well-edited but different domain text ("Featured-content" wikipedia pages) to a non-edited noisy text (live-journal blog posts from the "life" category). As we do not have gold-annotated data for these text genres, we analyze the few differences between the models, manually inspecting the instances on which the models disagree.

Table 8 describes our test corpora for this experiment. We applied the fully-lexicalized and the pruned ($k = 100$) anchored models described in Section 6.1 to these texts, and compared the chunking results. The results are presented in Table 9.

When moving outside of the canonic training

TEXT	#DIFF	PRUNED CORRECT	LEX CORRECT	BOTH WRONG
WSJ	13	**9**	4	0
Jaguar	45	20	20	7
FreeWill	118	**51**	38	29
LJ-Life	15	**8**	6	1

Table 9: Comparison of Models' performance on different text genres

corpus, the fully lexicalized model have no advantage over the heavily pruned one. On the contrary, the pruned models seem to have a small advantage in most cases (though it is hard to tell if the differences are significant). This is true even for texts in the very same domain, genre and editing guidelines as the training corpus was derived from.

8 Discussion

For all the sequence labeling tasks we analyzed, the anchored-SVM proved to be robust to feature pruning. The experiments support the claim that rare lexical features do not provide substantial information to the model, but instead play a role in maintaining separability. When this role is taken over by anchoring, we can obtain the same level of performance with very few robust lexical features. Yet, we cannot conclude that lexical information is not needed. There is a significant difference between the pruned and non-pruned models for the chunking task. We showed that this difference can be bridged to some extent by a binary feature relating to idiomatic word usage, and that the difference vanishes when testing outside of the annotated corpus. The high classification accuracies achieved with the heavily pruned anchored-SVM models sheds new light on the actual role of lexical features, and indicating that there is still a lot to be learned regarding the effective incorporation of lexical and semantic information into our models. It is our view that semantic knowledge should not be expected to be learned by inspection of raw lexical counts from an annotated text corpus, but instead collected from sources external to the annotated corpora – either based on a very large unannotated corpora, or on manually constructed lexical resources.

References

Daniel M. Bikel. 2004. Intricacies of collins' parsing model. *Computational Linguistics*, 30(4).

Richard P. Brent, 1973. *Algorithms for Minimization without Derivatives*, chapter 4. Prentice-Hall.

Claire Cardie and David Pierce. 1998. Error-driven pruning of treebank grammars for base noun phrase identification. In *ACL-1998*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proc of NAACL*.

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proc of EACL*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc of ACL*.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proc of EMNLP*.

Yoav Goldberg and Michael Elhadad. 2007. SVM Model Tampering and Anchored Learning: A Case Study in Hebrew. NP Chunking. In *ACL2007*.

Yoav Goldberg, Meni Adler, and Michael Elhadad. 2006. Noun Phrase Chunking in Hebrew: Influence of Lexical and Morphological Features. In *COLING/ACL2006*.

Hideki Isozaki and Hideto Kazawa. 2002. Efficient Support Vector Classifiers For Named Entity Recognition. In *COLING2002*.

Daisuke Kawahara and Kiyotaka Uchimoto. 2007. Miniamlly lexicalized dependency parsing. In *Proc of ACL (Short papers)*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*.

Yoshiaki Koshiba and Shigeo Abe. 2003. Comparison of L1 and L2 support vector machines. In *Proc. of the International Joint Conference on Neural Networks*, volume 3.

Taku Kudo and Yuji Matsumoto. 2000. Use of Support Vector Learning for Chunk Identification. In *CoNLL-2000*.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *NAACL '01*.

Mitch P. Marcus and Lance A. Ramshaw. 1995. Text Chunking Using Transformation-Based Learning. In *3rd ACL Workshop on Very Large Corpora*.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic cfg with latent annotations. In *Proc of ACL*.

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Joakim Nivre, Johan Hall, and Jens Nillson. 2006. MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *LREC2006*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc of ACL*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc of CONLL*.

Erik F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: chunking. In *CoNLL-2000*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proc of NAACL*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *CoNLL-2003*.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In *CoNLL-2002*.

Vladimir Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc.

Tong Zhang, Fred Damerau, and David Johnson. 2001. Text chunking using regularized winnow. In *Proc of ACL*.

Simple Coreference Resolution with Rich Syntactic and Semantic Features

Aria Haghighi and Dan Klein
Computer Science Division
UC Berkeley
{aria42, klein}@cs.berkeley.edu

Abstract

Coreference systems are driven by syntactic, semantic, and discourse constraints. We present a simple approach which completely modularizes these three aspects. In contrast to much current work, which focuses on learning and on the discourse component, our system is deterministic and is driven entirely by syntactic and semantic compatibility as learned from a large, unlabeled corpus. Despite its simplicity and discourse naivete, our system substantially outperforms all unsupervised systems and most supervised ones. Primary contributions include (1) the presentation of a simple-to-reproduce, high-performing baseline and (2) the demonstration that most remaining errors can be attributed to syntactic and semantic factors external to the coreference phenomenon (and perhaps best addressed by non-coreference systems).

1 Introduction

The resolution of entity reference is influenced by a variety of constraints. Syntactic constraints like the binding theory, the i-within-i filter, and appositive constructions restrict reference by configuration. Semantic constraints like selectional compatibility (e.g. a *spokesperson* can *announce* things) and subsumption (e.g. *Microsoft* is a *company*) rule out many possible referents. Finally, discourse phenomena such as salience and centering theory are assumed to heavily influence reference preferences. As these varied factors have given rise to a multitude of weak features, recent work has focused on how best to learn to combine them using models over reference structures (Culotta et al., 2007; Denis and Baldridge, 2007; Klenner and Ailloud, 2007).

In this work, we break from the standard view. Instead, we consider a vastly more modular system in which coreference is predicted from a deterministic function of a few rich features. In particular, we assume a three-step process. First, a self-contained syntactic module carefully represents syntactic structures using an augmented parser and extracts syntactic paths from mentions to potential antecedents. Some of these paths can be ruled in or out by deterministic but conservative syntactic constraints. Importantly, the bulk of the work in the syntactic module is in making sure the parses are correctly constructed and used, and this module's most important training data is a treebank. Second, a self-contained semantic module evaluates the semantic compatibility of headwords and individual names. These decisions are made from compatibility lists extracted from unlabeled data sources such as newswire and web data. Finally, of the antecedents which remain after rich syntactic and semantic filtering, reference is chosen to minimize tree distance.

This procedure is trivial where most systems are rich, and so does not need any supervised coreference data. However, it is rich in important ways which we argue are marginalized in recent coreference work. Interestingly, error analysis from our final system shows that its failures are far more often due to syntactic failures (e.g. parsing mistakes) and semantic failures (e.g. missing knowledge) than failure to model discourse phenomena or appropriately weigh conflicting evidence.

One contribution of this paper is the exploration of strong modularity, including the result that our system beats all unsupervised systems and approaches the state of the art in supervised ones. Another contribution is the error analysis result that, even with substantial syntactic and semantic richness, the path to greatest improvement appears to be to further improve the syntactic and semantic modules. Finally, we offer our approach as a very strong, yet easy to implement, baseline. We make no claim that learning to reconcile disparate features in a joint model offers no benefit, only that it must not be pursued to the exclusion of rich, non-reference analysis.

2 Coreference Resolution

In coreference resolution, we are given a document which consists of a set of mentions; each

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1152–1161,
Singapore, 6-7 August 2009.

mention is a phrase in the document (typically an NP) and we are asked to cluster mentions according to the underlying referent entity. There are three basic mention types: proper (*Barack Obama*), nominal (*president*), and pronominal (*he*).[1] For comparison to previous work, we evaluate in the setting where mention boundaries are given at test time; however our system can easily annotate reference on all noun phrase nodes in a parse tree (see Section 3.1.1).

2.1 Data Sets

In this work we use the following data sets:

Development: (see Section 3)

- **ACE2004-ROTH-DEV**: Dev set split of the ACE 2004 training set utilized in Bengston and Roth (2008). The ACE data also annotates pre-nominal mentions which we map onto nominals. 68 documents and 4,536 mentions.

Testing: (see Section 4)

- **ACE2004-CULOTTA-TEST**: Test set split of the ACE 2004 training set utilized in Culotta et al. (2007) and Bengston and Roth (2008). Consists of 107 documents.[2]
- **ACE2004-NWIRE**: ACE 2004 Newswire set to compare against Poon and Domingos (2008). Consists of 128 documents and 11,413 mentions; intersects with the other ACE data sets.
- **MUC-6-TEST**: MUC6 formal evaluation set consisting of 30 documents and 2,068 mentions.

Unlabeled: (see Section 3.2)

- **BLIPP**: 1.8 million sentences of newswire parsed with the Charniak (2000) parser. No labeled coreference data; used for mining semantic information.
- **WIKI**: 25k articles of English Wikipedia abstracts parsed by the Klein and Manning (2003) parser.[3] No labeled coreference data; used for mining semantic information.

[1] Other mention types exist and are annotated (such as pre-nominal), which are treated as nominals in this work.

[2] The evaluation set was not made available to non-participants.

[3] Wikipedia abstracts consist of roughly the first paragraph of the corresponding article

2.2 Evaluation

We will present evaluations on multiple coreference resolution metrics, as no single one is clearly superior:

- Pairwise F1: precision, recall, and F1 over all pairs of mentions in the same entity cluster. Note that this over-penalizes the merger or separation of clusters quadratically in the size of the cluster.
- b^3 (Amit and Baldwin, 1998): For each mention, form the intersection between the predicted cluster and the true cluster for that mention. The precision is the ratio of the intersection and the true cluster sizes and recall the ratio of the intersection to the predicted sizes; F1 is given by the harmonic mean over precision and recall from all mentions.
- MUC (Vilain et al., 1995): For each true cluster, compute the number of predicted clusters which need to be merged to cover the true cluster. Divide this quantity by true cluster size minus one. Recall is given by the same procedure with predicated and true clusters reversed.[4]
- CEAF (Luo, 2005): For a similarity function between predicted and true clusters, CEAF scores the best match between true and predicted clusters using this function. We use the ϕ_3 similarity function from Luo (2005).

3 System Description

In this section we develop our system and report developmental results on ACE2004-ROTH-DEV (see Section 2.1); we report pairwise F1 figures here, but report on many more evaluation metrics in Section 4. At a high level, our system resembles a pairwise coreference model (Soon et al., 1999; Ng and Cardie, 2002; Bengston and Roth, 2008); for each mention m_i, we select either a single-best antecedent amongst the previous mentions $m_1, \ldots, m_{i-1}$, or the NULL mention to indicate the underlying entity has not yet been evoked. Mentions are linearly ordered according to the position of the mention head with ties being broken by the larger node coming first.

[4] The MUC measure is problematic when the system predicts many more clusters than actually exist (Luo, 2005; Finkel and Manning, 2008); also, singleton clusters do not contribute to evaluation.

While much research (Ng and Cardie, 2002; Culotta et al., 2007; Haghighi and Klein, 2007; Poon and Domingos, 2008; Finkel and Manning, 2008) has explored how to reconcile pairwise decisions to form coherent clusters, we simply take the transitive closure of our pairwise decision (as in Ng and Cardie (2002) and Bengston and Roth (2008)) which can and does cause system errors.

In contrast to most recent research, our pairwise decisions are not made with a learned model which outputs a probability or confidence, but instead for each mention m_i, we select an antecedent amongst $m_1, \ldots, m_{i-1}$ or the NULL mention as follows:

- **Syntactic Constraint:** Based on syntactic configurations, either force or disallow coreference between the mention and an antecedent. Propagate this constraint (see Figure 4).
- **Semantic/Syntactic Filter**: Filter the remaining possible antecedents based upon compatibility with the mention (see Figure 2).
- **Selection**: Select the 'closest' mention from the set of remaining possible antecedents (see Figure 1) or the NULL antecedent if empty.

Initially, there is no syntactic constraint (improved in Section 3.1.3), the antecedent compatibility filter allows proper and nominal mentions to corefer only with mentions that have the same head (improved in Section 3.2), and pronouns have no compatibility constraints (improved in Section 3.1.2). Mention heads are determined by parsing the given mention span with the Stanford parser (Klein and Manning, 2003) and using the Collins head rules (Collins, 1999); Poon and Domingos (2008) showed that using syntactic heads strongly outperformed a simple rightmost headword rule. The mention type is determined by the head POS tag: proper if the head tag is NNP or NNPS, pronoun if the head tag is PRP, PRP$, WP, or WP$, and nominal otherwise.

For the selection phase, we order mentions $m_1, \ldots, m_{i-1}$ according to the position of the head word and select the closest mention that remains after constraint and filtering are applied. This choice reflects the intuition of Grosz et al. (1995) that speakers only use pronominal mentions when there are not intervening compatible

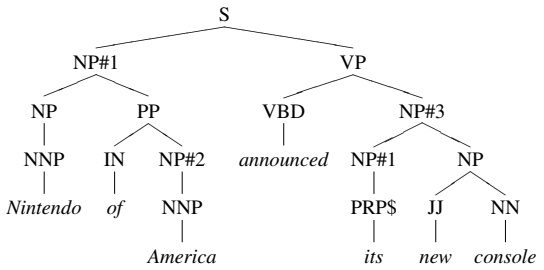


Figure 1: Example sentence where closest tree distance between mentions outperforms raw distance. For clarity, each mention NP is labeled with the underlying entity id.

mentions. This system yields a rather low 48.9 pairwise F1 (see BASE-FLAT in Table 2). There are many, primarily recall, errors made choosing antecedents for all mention types which we will address by adding syntactic and semantic constraints.

3.1 Adding Syntactic Information

In this section, we enrich the syntactic representation and information in our system to improve results.

3.1.1 Syntactic Salience

We first focus on fixing the pronoun antecedent choices. A common error arose from the use of mention head distance as a poor proxy for discourse salience. For instance consider the example in Figure 1, the mention *America* is closest to *its* in flat mention distance, but syntactically *Nintendo of America* holds a more prominent syntactic position relative to the pronoun which, as Hobbs (1977) argues, is key to discourse salience.

Mapping Mentions to Parse Nodes: In order to use the syntactic position of mentions to determine anaphoricity, we must associate each mention in the document with a parse tree node. We parse all document sentences with the Stanford parser, and then for each evaluation mention, we find the largest-span NP which has the previously determined mention head as its head.[5] Often, this results in a different, typically larger, mention span than annotated in the data.

Now that each mention is situated in a parse tree, we utilize the length of the shortest tree path between mentions as our notion of distance. In

[5]If there is no NP headed by a given mention head, we add an NP over just that word.

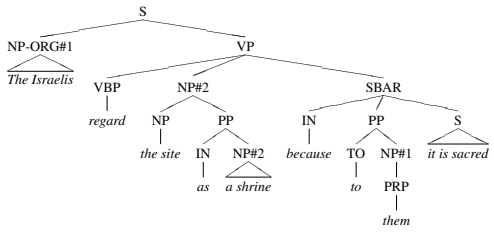


Figure 2: Example of a coreference decision fixed by agreement constraints (see Section 3.1.2). The pronoun *them* is closest to *the site* mention, but has an incompatible number feature with it. The closest (in tree distance, see Section 3.1.1) compatible mention is *The Israelis*, which is correct

particular, this fixes examples such as those in Figure 1 where the true antecedent has many embedded mentions between itself and the pronoun. This change by itself yields 51.7 pairwise F1 (see BASE-TREE in Table 2), which is small overall, but reduces pairwise pronoun antecedent selection error from 51.3% to 42.5%.

3.1.2 Agreement Constraints

We now refine our compatibility filtering to incorporate simple agreement constraints between coreferent mentions. Since we currently allow proper and nominal mentions to corefer only with matching head mentions, agreement is only a concern for pronouns. Traditional linguistic theory stipulates that coreferent mentions must agree in number, person, gender, and entity type (e.g. animacy). Here, we implement person, number and entity type agreement.[6]

A number feature is assigned to each mention deterministically based on the head and its POS tag. For entity type, we use NER labels. Ideally, we would like to have information about the entity type of each referential NP, however this information is not easily obtainable. Instead, we opt to utilize the Stanford NER tagger (Finkel et al., 2005) over the sentences in a document and annotate each NP with the NER label assigned to that mention head. For each mention, when its NP is assigned an NER label we allow it to only be compatible with that NER label.[7] For pronouns, we deterministically assign a set of compatible NER values (e.g. personal pronouns can only be a PERSON, but *its* can be an ORGANIZATION or LOCATION). Since the NER tagger typically does not label non-proper NP heads, we have no NER compatibility information for nominals.

[6]Gender agreement, while important for general coreference resolution, did not contribute to the errors in our largely newswire data sets.

[7]Or allow it to be compatible with all NER labels if the NER tagger doesn't predict a label.

gore	president	florida	state
bush	governor	lebanese	territory
nation	people	arafat	leader
inc.	company	aol	company
nation	country	assad	president

Table 1: Most common recall (missed-link) errors amongst non-pronoun mention heads on our development set. Detecting compatibility requires semantic knowledge which we obtain from a large corpus (see Section 3.2).

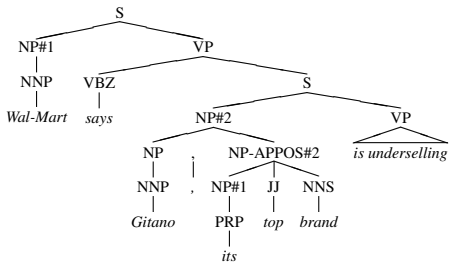


Figure 4: Example of interaction between the appositive and i-within-i constraint. The i-within-i constraint disallows coreference between parent and child NPs unless the child is an appositive. Hashed numbers indicate ground truth but are not in the actual trees.

We incorporate agreement constraints by filtering the set of possible antecedents to those which have compatible number and NER types with the target mention. This yields 53.4 pairwise F1, and reduces pronoun antecedent errors to 42.5% from 34.4%. An example of the type of error fixed by these agreement constraints is given by Figure 2.

3.1.3 Syntactic Configuration Constraints

Our system has so far focused only on improving pronoun anaphora resolution. However, a plurality of the errors made by our system are amongst non-pronominal mentions.[8] We take the approach that in order to align a non-pronominal mention to an antecedent without an identical head, we require evidence that the mentions are compatible.

Judging compatibility of mentions generally requires semantic knowledge, to which we return later. However, some syntactic configurations

[8]There are over twice as many nominal mentions in our development data as pronouns.

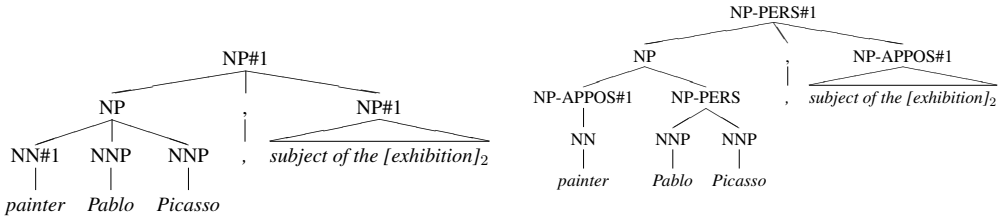


Figure 3: NP structure annotation: In (a) we have the raw parse from the Klein and Manning (2003) parser with the mentions annotated by entity. In (b), we demonstrate the annotation we have added. NER labels are added to all NP according to the NER label given to the head (see Section 3.1.1). Appositive NPs are also annotated. Hashes indicate forced coreferent nodes

guarantee coreference. The one exploited most in coreference work (Soon et al., 1999; Ng and Cardie, 2002; Luo et al., 2004; Culotta et al., 2007; Poon and Domingos, 2008; Bengston and Roth, 2008) is the appositive construction. Here, we represent apposition as a syntactic feature of an NP indicating that it is coreferent with its parent NP (e.g. it is an exception to the i-within-i constraint that parent and child NPs *cannot* be coreferent). We deterministically mark a node as NP-APPOS (see Figure 3) when it is the third child in of a parent NP whose expansion begins with (NP , NP), and there is not a conjunction in the expansion (to avoid marking elements in a list as appositive).

Role Appositives: During development, we discovered many errors which involved a variant of appositives which we call 'role appositives' (see *painter* in Figure 3), where an NP modifying the head NP describes the role of that entity (typically a person entity). There are several challenges to correctly labeling these role NPs as being appositives. First, the NPs produced by Treebank parsers are flat and do not have the required internal structure (see Figure 3(a)). While fully solving this problem is difficult, we can heuristically fix many instances of the problem by placing an NP around maximum length sequences of NNP tags or NN (and JJ) tags within an NP; note that this will fail for many constructions such as *U.S. President Barack Obama*, which is analyzed as a flat sequence of proper nouns. Once this internal NP structure has been added, whether the NP immediately to the left of the head NP is an appositive depends on the entity type. For instance, *Rabbi Ashi* is an apposition but *Iranian army* is not. Again, a full solution would require its own model, here we mark as appositions any NPs immediately to the left of a head child NP where the head child NP is identified as a person by the NER tagger.[9]

We incorporate NP appositive annotation as a constraint during filtering. Any mention which corresponds to an appositive node has its set of possible antecedents limited to its parent. Along with the appositive constraint, we implement the i-within-i constraint that any non-appositive NP cannot be be coreferent with its parent; this constraint is then propagated to any node its parent is forced to agree with. The order in which these constraints are applied is important, as illustrated by the example in Figure 4: First the list of possible antecedents for the appositive NP is constrained to only its parent. Now that all appositives have been constrained, we apply the i-within-i constraint, which prevents *its* from having the NP headed by *brand* in the set of possible antecedents, and by propagation, also removes the NP headed by *Gitano*. This leaves the NP *Wal-Mart* as the closest compatible mention.

Adding these syntactic constraints to our system yields 55.4 F1, a fairly substantial improvement, but many recall errors remain between mentions with differing heads. Resolving such cases will require external semantic information, which we will automatically acquire (see Section 3.2).

Predicate Nominatives: Another syntactic constraint exploited in Poon and Domingos (2008) is the predicate nominative construction, where the object of a copular verb (forms of the verb *be*) is constrained to corefer with its subject (e.g. *Microsoft is a company in Redmond*). While much less frequent than appositive configurations (there are only 17 predicate nominatives in our devel-

[9] Arguably, we could also consider right modifying NPs (e.g., *[Microsoft [Company]$_1$]$_1$*) to be role appositive, but we do not do so here.

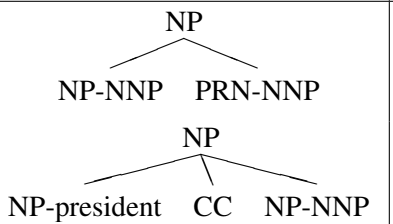
Path	Example
	*America Online Inc. (AOL)* *[President and C.E.O] Bill Gates*

Figure 5: Example paths extracted via semantic compatibility mining (see Section 3.2) along with example instantiations. In both examples the left child NP is coreferent with the rightmost NP. Each category in the interior of the tree path is annotated with the head word as well as its subcategorization. The examples given here collapse multiple instances of extracted paths.

opment set), predicate nominatives are another highly reliable coreference pattern which we will leverage in Section 3.2 to mine semantic knowledge. As with appositives, we annotate object predicate-nominative NPs and constrain coreference as before. This yields a minor improvement to 55.5 F1.

3.2 Semantic Knowledge

While appositives and related syntactic constructions can resolve some cases of non-pronominal reference, most cases require semantic knowledge about the various entities as well as the verbs used in conjunction with those entities to disambiguate references (Kehler et al., 2008).

However, given a semantically compatible mention head pair, say *AOL* and *company*, one might expect to observe a reliable appositive or predicative-nominative construction involving these mentions somewhere in a large corpus. In fact, the Wikipedia page for *AOL*[10] has a predicate-nominative construction which supports the compatibility of this head pair: *AOL LLC (formerly America Online) is an American global Internet services and media company operated by Time Warner.*

In order to harvest compatible head pairs, we utilize our BLIPP and WIKI data sets (see Section 2), and for each noun (proper or common) and pronoun, we assign a maximal NP mention node for each nominal head as in Section 3.1.1; we then annotate appositive and predicate-nominative NPs as in Section 3.1.3. For any NP which is annotated as an appositive or predicate-nominative, we extract the head pair of that node and its constrained antecedent.

[10] http://en.wikipedia.org/wiki/AOL

The resulting set of compatible head words, while large, covers a little more than half of the examples given in Table 1. The problem is that these highly-reliable syntactic configurations are too sparse and cannot capture all the entity information present. For instance, the first sentence of Wikipedia abstract for *Al Gore* is:

> *Albert Arnold "Al" Gore, Jr. is an American environmental activist who served as the 45th Vice President of the United States from 1993 to 2001 under President Bill Clinton.*

The required lexical pattern *X who served as Y* is a general appositive-like pattern that almost surely indicates coreference. Rather than opt to manually create a set of these coreference patterns as in Hearst (1992), we instead opt to automatically extract these patterns from large corpora as in Snow et al. (2004) and Phillips and Riloff (2007). We take a simple bootstrapping technique: given a set of mention pairs extracted from appositives and predicate-nominative configurations, we extract counts over tree fragments between nodes which have occurred in this set of head pairs (see Figure 5); the tree fragments are formed by annotating the internal nodes in the tree path with the head word and POS along with the subcategorization. We limit the paths extracted in this way in several ways: paths are only allowed to go between adjacent sentences and have a length of at most 10. We then filter the set of paths to those which occur more than a hundred times and with at least 10 distinct seed head word pairs.

The vast majority of the extracted fragments are variants of traditional appositives and predicate-nominatives with some of the structure of the NPs

System	MUC P	MUC R	MUC F1	b^3 P	b^3 R	b^3 F1	Pairwise P	Pairwise R	Pairwise F1	CEAF P	CEAF R	CEAF F1
ACE2004-ROTH-DEV												
BASIC-FLAT	73.5	66.8	70.0	80.6	68.6	74.1	63.6	39.7	48.9	68.4	68.4	68.4
BASIC-TREE	75.8	68.9	72.2	81.9	69.9	75.4	65.6	42.7	51.7	69.8	69.8	69.8
+SYN-COMPAT	77.8	68.5	72.9	84.1	69.7	76.2	71.0	43.1	53.4	69.8	69.8	69.8
+SYN-CONSTR	78.3	70.5	74.2	84.0	71.0	76.9	71.3	45.4	55.5	70.8	70.8	70.8
+SEM-COMPAT	77.9	74.1	75.9	81.8	74.3	77.9	68.2	51.2	58.5	72.5	72.5	72.5
ACE2004-CULOTTA-TEST												
BASIC-FLAT	68.6	60.9	64.5	80.3	68.0	73.6	57.1	30.5	39.8	66.5	66.5	66.5
BASIC-TREE	71.2	63.2	67.0	81.6	69.3	75.0	60.1	34.5	43.9	67.9	67.9	67.9
+SYN-COMPAT	74.6	65.2	69.6	84.2	70.3	76.6	66.7	37.2	47.8	69.2	69.2	69.2
+SYN-CONSTR	74.3	66.4	70.2	83.6	71.0	76.8	66.4	38.0	48.3	69.6	69.6	69.6
+SEM-COMPAT	74.8	**77.7**	**79.6**	79.6	**78.5**	79.0	**57.5**	57.6	57.5	**73.3**	**73.3**	**73.3**
Supervised Results												
Culotta et al. (2007)	-	-	-	86.7	73.2	79.3	-	-	-	-	-	-
Bengston and Roth (2008)	**82.7**	69.9	75.8	**88.3**	74.5	**80.8**	55.4	**63.7**	**59.2**	-	-	-
MUC6-TEST												
+SEM-COMPAT	**87.2**	**77.3**	**81.9**	84.7	**67.3**	**75.0**	**80.5**	**57.8**	**67.3**	72.0	72.0	72.0
Unsupervised Results												
Poon and Domingos (2008)	83.0	75.8	79.2	-	-	-	63.0	57.0	60.0	-	-	-
Supervised Results												
Finkel and Manning (2008)	89.7	55.1	68.3	**90.9**	49.7	64.3	74.1	37.1	49.5	-	-	-
ACE2004-NWIRE												
+SEM-COMPAT	**77.0**	**75.9**	**76.5**	79.4	74.5	76.9	**66.9**	**49.2**	**56.7**	71.5	71.5	71.5
Unsupervised Results												
Poon and Domingos (2008)	71.3	70.5	70.9	-	-	-	62.6	38.9	48.0	-	-	-

Table 2: Experimental Results (See Section 4): When comparisons between systems are presented, the largest result is bolded. The CEAF measure has equal values for precision, recall, and F1.

specified. However there are some tree fragments which correspond to the novel coreference patterns (see Figure 5) of parenthetical alias as well as conjunctions of roles in NPs.

We apply our extracted tree fragments to our BLIPP and WIKI data sets and extract a set of compatible word pairs which match these fragments; these words pairs will be used to relax the semantic compatibility filter (see the start of the section); mentions are compatible with prior mentions with the same head or with a semantically compatible head word. This yields 58.5 pairwise F1 (see SEM-COMPAT in Table 2) as well as similar improvements across other metrics.

By and large the word pairs extracted in this way are correct (in particular we now have coverage for over two-thirds of the head pair recall errors from Table 1.) There are however word-pairs which introduce errors. In particular city-state constructions (e.g. *Los Angeles, California*) appears to be an appositive and incorrectly allows our system to have *angeles* as an antecedent for *california*. Another common error is that the % symbol is made compatible with a wide variety of common nouns in the financial domain.

4 Experimental Results

We present formal experimental results here (see Table 2). We first evaluate our model on the ACE2004-CULOTTA-TEST dataset used in the state-of-the-art systems from Culotta et al. (2007) and Bengston and Roth (2008). Both of these systems were supervised systems discriminatively trained to maximize b^3 and used features from many different structured resources including WordNet, as well as domain-specific features (Culotta et al., 2007). Our best b^3 result of 79.0 is broadly in the range of these results. We should note that in our work we use neither the gold mention types (we do not model pre-nominals separately) nor do we use the gold NER tags which Bengston and Roth (2008) does. Across metrics, the syntactic constraints and semantic compatibility components contribute most to the overall final result.

On the MUC6-TEST dataset, our system outper-

	PROPER	NOMINAL	PRONOUN	NULL	TOTAL
PROPER	21/451	8/20	-	72/288	101/759
NOMINAL	16/150	99/432	-	158/351	323/933
PRONOUN	29/149	60/128	15/97	1/2	105/376

Table 3: Errors for each type of antecedent decision made by the system. Each row is a mention type and the column the predicted mention type antecedent. The majority of errors are made in the NOMINAL category.

forms both Poon and Domingos (2008) (an unsupervised Markov Logic Network system which uses explicit constraints) and Finkel and Manning (2008) (a supervised system which uses ILP inference to reconcile the predictions of a pairwise classifier) on all comparable measures.[11] Similarly, on the ACE2004-NWIRE dataset, we also outperform the state-of-the-art unsupervised system of Poon and Domingos (2008).

Overall, we conclude that our system outperforms state-of-the-art unsupervised systems[12] and is in the range of the state-of-the art systems of Culotta et al. (2007) and Bengston and Roth (2008).

5 Error Analysis

There are several general trends to the errors made by our system. Table 3 shows the number of pairwise errors made on MUC6-TEST dataset by mention type; note these errors are not equally weighted in the final evaluations because of the transitive closure taken at the end. The most errors are made on nominal mentions with pronouns coming in a distant second. In particular, we most frequently say a nominal is NULL when it has an antecedent; this is typically due to not having the necessary semantic knowledge to link a nominal to a prior expression.

In order to get a more thorough view of the cause of pairwise errors, we examined 20 random errors made in aligning each mention type to an antecedent. We categorized the errors as follows:

- SEM. COMPAT: Missing information about the compatibility of two words e.g. *pay* and *wage*. For pronouns, this is used to mean that we incorrectly aligned a pronoun to a mention with which it is not semantically compatible (e.g. *he* aligned to *board*).
- SYN. COMPAT: Error in assigning linguistic features of nouns for compatibility with pronouns (e.g. disallowing *they* to refer to *team*).
- HEAD: Errors involving the assumption that mentions with the same head are always compatible. Includes modifier and specificity errors such as allowing *Lebanon* and *Southern Lebanon* to corefer. This also includes errors of definiteness in nominals (e.g. *the people in the room* and *Chinese people*). Typically, these errors involve a combination of missing syntactic and semantic information.
- INTERNAL NP: Errors involving lack of internal NP structure to mark role appositives (see Section 3.1.3).
- PRAG. / DISC.: Errors where discourse salience or pragmatics are needed to disambiguate mention antecedents.
- PROCESS ERROR: Errors which involved a tokenization, parse, or NER error.

The result of this error analysis is given in Table 4; note that a single error may be attributed to more than one cause. Despite our efforts in Section 3 to add syntactic and semantic information to our system, the largest source of error is still a combination of missing semantic information or annotated syntactic structure rather than the lack of discourse or salience modeling.

Our error analysis suggests that in order to improve the state-of-the-art in coreference resolution, future research should consider richer syntactic and semantic information than typically used in current systems.

6 Conclusion

Our approach is not intended as an argument against the more complex, discourse-focused approaches that typify recent work. Instead, we note that rich syntactic and semantic processing vastly reduces the need to rely on discourse effects or evidence reconciliation for reference resolution. Indeed, we suspect that further improving the syntactic and semantic modules in our system may produce greater error reductions than any other

[11] Klenner and Ailloud (2007) took essentially the same approach but did so on non-comparable data.

[12] Poon and Domingos (2008) outperformed Haghighi and Klein (2007). Unfortunately, we cannot compare against Ng (2008) since we do not have access to the version of the ACE data used in their evaluation.

Mention Type	SEM. COMPAT	SYN. COMPAT	HEAD	INTENAL NP	PRAG / DISC.	PROCESS ERROR	OTHER	Comment
NOMINAL	7	-	5	6	2	2	1	2 general appos. patterns
PRONOUN	6	3	-	6	3	3	3	2 cataphora
PROPER	6	-	3	4	4	4	1	

Table 4: Error analysis on ACE2004-CULOTTA-TEST data by mention type. The dominant errors are in either semantic or syntactic compatibility of mentions rather than discourse phenomena. See Section 5.

route forward. Of course, a system which is rich in all axes will find some advantage over any simplified approach.

Nonetheless, our coreference system, despite being relatively simple and having no tunable parameters or complexity beyond the non-reference complexity of its component modules, manages to outperform state-of-the-art unsupervised coreference resolution and be broadly comparable to state-of-the-art supervised systems.

References

B. Amit and B. Baldwin. 1998. Algorithms for scoring coreference chains. In *MUC7*.

Eric Bengston and Dan Roth. 2008. Understanding the value of features for corefernce resolution. In *Empirical Methods in Natural Language Processing*.

E. Charniak. 2000. Maximum entropy inspired parser. In *North American Chapter of the Association of Computational Linguistics (NAACL)*.

Mike Collins. 1999. Head-driven statistical models for natural language parsing.

A Culotta, M Wick, R Hall, and A McCallum. 2007. First-order probabilistic models for coreference resolution. In *NAACL-HLT*.

Pascal Denis and Jason Baldridge. 2007. Global, Joint Determination of Anaphoricity and Coreference Resolution using Integer Programming. In *HLT-NAACL*.

Jenny Finkel and Christopher Manning. 2008. Enforcing transitivity in coreference resolution. In *Association of Computational Linguists (ACL)*.

Jenny Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modelling the local coherence of discourse.

Aria Haghighi and Dan Klein. 2007. Unsupervised coreference resolution in a nonparametric bayesian model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Conference on Natural Language Learning (COLING)*.

J. R. Hobbs. 1977. Resolving pronoun references. *Lingua*.

Andrew Kehler, Laura Kertz, Hannah Rohde, and Jeffrey Elman. 2008. Coherence and coreference revisited.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *Association of Computational Linguists (ACL)*.

Manfred Klenner and Etienne Ailloud. 2007. Optimization in coreference resolution is not needed: A nearly-optimal algorithm with intensional constraints. In *Recent Advances in Natural Language Processing*.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A Mention-Synchronous Coreference Resolution Algorithm Based on the Bell Tree. In *Association of Computational Linguists*.

X Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*.

Vincent Ng and Claire Cardie. 2002. Improving Machine Learning Approaches to Coreference Resolution. In *Association of Computational Linguists*.

Vincent Ng. 2008. Unsupervised models of coreference resolution. In *EMNLP*.

W. Phillips and E. Riloff. 2007. Exploiting role-identifying nouns and expressions for information extraction. In *Recent Advances in Natural Language Processing (RANLP)*.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov Logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*.

R. Snow, D. Jurafsky, and A. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Neural Information Processing Systems (NIPS)*.

W.H. Soon, H. T. Ng, and D. C. Y. Lim. 1999. A machine learning approach to coreference resolution of noun phrases.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *MUC-6*.

Descriptive and Empirical Approaches to Capturing Underlying Dependencies among Parsing Errors

Tadayoshi Hara[1] **Yusuke Miyao**[1] **Jun'ichi Tsujii**[1,2,3]
[1]Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, 113-0033, JAPAN
[2]School of Computer Science, University of Manchester
[3]NaCTeM (National Center for Text Mining)
{harasan,yusuke,tsujii}@is.s.u-tokyo.ac.jp

Abstract

In this paper, we provide descriptive and empirical approaches to effectively extracting underlying dependencies among parsing errors. In the descriptive approach, we define some combinations of error patterns and extract them from given errors. In the empirical approach, on the other hand, we re-parse a sentence with a target error corrected and observe errors corrected together. Experiments on an HPSG parser show that each of these approaches can clarify the dependencies among individual errors from each point of view. Moreover, the comparison between the results of the two approaches shows that combining these approaches can achieve a more detailed error analysis.

1 Introduction

For any kind of technology, analyzing causes of errors given by a system is a very helpful process for improving its performance. In recent sophisticated parsing technologies, the process has taken on more and more important roles since critical ideas for parsing performance have already been introduced and the researches are now focusing on exploring the rest of the pieces for making additional improvements.

In most cases for parsers' error analysis, researchers associate output errors with failures in handling certain linguistic phenomena and attempt to avoid them by adding or modifying corresponding settings of their parsers. However, such an analysis cannot been done so smoothly since parsing errors sometimes depend on each other and the underlying dependencies behind superficial phenomena cannot be captured easily.

In this paper, we propose descriptive and empirical approaches to effective extraction of dependencies among parsing errors and engage in a deeper error analysis with them. In our descriptive approach, we define various combinations of error patterns as organized error phenomena on the basis of linguistic knowledge, and then extract such combinations from given errors. In our empirical approach, on the other and, we re-parse a sentence under the condition where a target error is corrected, and errors which are additionally corrected are regarded as dependent errors. By capturing dependencies among parsing errors through systematic approaches, we can effectively collect errors which are related to the same linguistic properties.

In the experiments, we applied both of our approaches to an HPSG parser *Enju* (Miyao and Tsujii, 2005; Ninomiya et al., 2006), and then evaluated the obtained error classes. After examining the individual approaches, we explored the combination of them.

2 Parser and its evaluation

A parser is a system which interprets structures of given sentences from some grammatical or in some cases semantical viewpoints, and interpreted structures are utilized as essential information for various natural language tasks such as information extraction, machine translation, and so on. In most cases, an output structure of a parser is based on a certain grammatical framework such as CFG, CCG (Steedman, 2000), LFG (Kaplan and Bresnan, 1995) or HPSG (Pollard and Sag, 1994). Since such a framework can usually produce more than one probable structure for a sentence, a parser

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1162–1171,
Singapore, 6-7 August 2009.

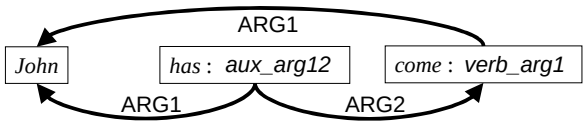


Figure 1: Predicate argument relations

Abbr.	Full	Abbr.	Full
aux	auxiliary	*lgs*	logical subject
verb	verb	*coord*	coordination
prep	prepositional	*conj*	conjunction
det	determiner	*_arg*N_1...	take argument(s) (N_1th, ...)
adj	adjunction		
app	apposition	*_mod*	modify a word
relative	relative		

Table 1: Descriptions for predicate types

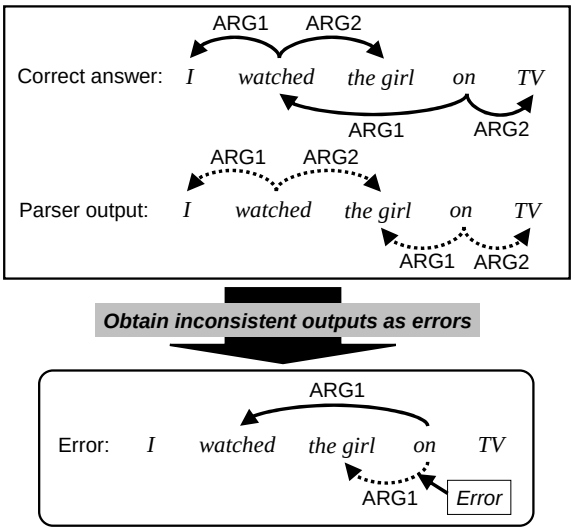


Figure 2: An example of parsing errors

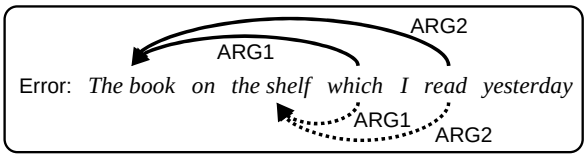


Figure 3: Co-occurring parsing errors

often utilizes some kind of disambiguation model for choosing the best one.

While various parsers take different manners in capturing linguistic phenomena based on their frameworks, they are at least required to obtain some kinds of relations between the words in sentences. On the basis of the requirements, a parser is usually evaluated on how correctly it gives intended linguistic relations. "Predicate argument relation" is one of the most common evaluation measurements for a parser since it is a very fundamental linguistic behavior and is less dependent on parser systems. This measure divides linguistic structural phenomena in a sentence into minimal predicative events. In one predicate argument relation, a word which represents an event (predicate) takes some words as participants (arguments). Although no fixed formulation exists for the relations, there are to a large extent common conceptions for them based on linguistic knowledge among researchers.

Figure 1 shows an example of predicate argument relations given by *Enju*. In the sentence "*John has come.*", "*has*" is a predicate of type "*aux_arg12*" and takes "*John*" and "*come*" as the first and second arguments. "*come*" is also a predicate of the type "*verb_arg1*" and takes "*John*" as the first and the only argument. In this formalism, each predicate type is represented as a combination of "the grammatical nature of a word" and "the arguments which it takes," which are represented by the descriptions in Table 1. "*aux_arg12*" in Figure 1 indicates that it is an auxiliary word and takes two arguments "ARG1" and "ARG2."

In order to improve the performance of a parser, analyzing parsing errors is very much worth the effort. Since the errors are output according to a given evaluation measurement such as "predicate argument relation," we researchers carefully explore them and infer the linguistic phenomena which cause the erroneous outputs. Figure 2 shows an example of parsing errors for sentence "*I watched the girl on TV.*" Note that the errors are based on predicate argument relations as shown above and that the predicate types are abbreviated in this figure. When we focus on the error output, we can observe that "ARG1" of predicate "*on*" was mistaken by the parser. In this case, "ARG1" represents a modifiee of the preposition, and we then conclude that the ill attachment of a prepositional phrase caused this error. By continuing such error analysis, weak points of the parser are revealed and can be useful clues for further improvements.

However, in most researches on parsing technologies, error analysis has been limited to narrow and shallow explorations since there are various dependencies behind erroneous outputs. In Figure 3, for example, two errors were given: wrong outputs for "ARG1" of "*which*" and "ARG2" of "*read*." Both of these two errors originated from the fact that the relative clause took a wrong antecedent "*the shelf*." In this sentence, the former

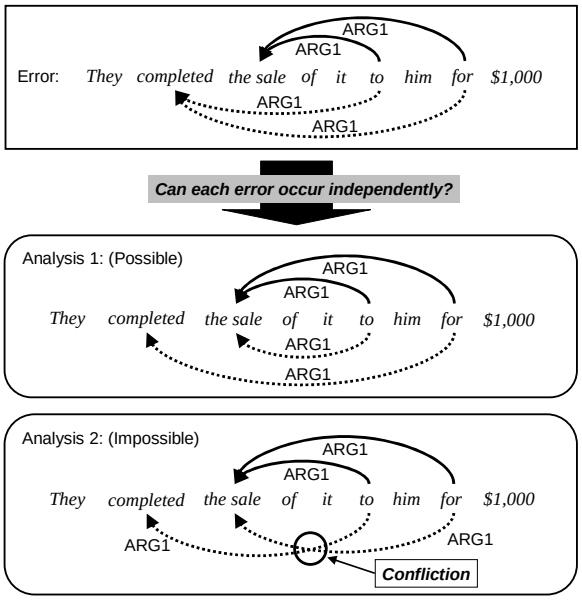


Figure 4: Sketch of error propagation

Erroneous phenomena	Matched patterns
[Argument selection]	
Prepositional attachment	ARG1 of *prep_arg*
Adjunction attachment	ARG1 of *adj_arg*
Conjunction attachment	ARG1 of *conj_arg*
Head selection for noun phrase	ARG1 of *det_arg*
Coordination	ARG1/2 of *coord_arg*
[Predicate type selection]	
Preposition/Adjunction	*prep_arg* / *adj_arg*
Gerund acts as modifier/not	*verb_mod_arg* / *verb_arg*
Coordination/conjunction	*coord_arg* / *conj_arg*
# of arguments for preposition	*prep_argX* / *prep_argY* $(X \neq Y)$
Adjunction/adjunctive noun	*adj_arg* / *noun_arg*
[More structural errors]	
To-infinitive for modifier/argument of verb	see Figure 7
Subject for passive sentence or not	see Figure 8
[Others]	
Comma	any error around ","
Relative clause attachment	see Figure 9

Table 2: Patterns defined for descriptive approach

"ARG1" directly corresponds to the antecedent while the latter "ARG2" indirectly referred to the same antecedent as the object of the verb "*read*." The two predicate argument relations thus took the same word as their common arguments, and therefore the two errors co-occurred.

On the other hand, one-way inductive relations also exist among errors. In Figure 4, "ARG1" of "*for*" and "*to*" were mistaken by a parser. We can know that each of the errors was caused by an ill attachment of a prepositional phrase with the same analysis as shown in Figure 2. What is important in this example is the manner in their occurrences. The former error can appear by itself (Analysis 1) while the latter cannot because of the structural conflict with the former error (Analysis 2). The appearance of the latter error thus induces that of the former error. In error analysis, we have to correctly capture such various relations, which leads us to a costly and less rewarding analysis.

In order to make advancements on this problem, we propose two types of approaches to realizing a deeper error analysis on parsing. In the experiments, we examine our approaches for actual errors which are given by the HPSG parser *Enju* (Miyao and Tsujii, 2005; Ninomiya et al., 2006). *Enju* was developed for capturing detailed syntactic or semantic properties and relations for a sentence with an HPSG framework (Pollard and Sag, 1994). In this research, we focus on error analysis based on predicate argument relations, and in the experiments with *Enju*, utilize the relations which are represented in parsed tree structures.

3 Two approaches for error analysis

In this section, we propose two approaches for error analysis which enable us to capture underlying dependencies among parsing errors. Our descriptive approach matches the patterns of error combinations with given parsing errors and collects matched erroneous participants. Our empirical approach, on the other hand, detects co-occurring errors by re-parsing a sentence under a situation where each of the errors is forcibly corrected.

3.1 Descriptive approach

Our descriptive approach for capturing dependencies among parsing errors is to extract certain representative structures of errors and collect the errors which involve them. Parsing errors have a tendency to occur with certain patterns of structures representing linguistic phenomena. We first define such patterns through observations with a part of error outputs, and then match them with the rest.

Table 2 summarizes the patterns for erroneous phenomena which we defined for matching in the experiments. In the table, the patterns for 14 phenomena are given and classified into four types according to their matching manners. Each of the patterns for "Argument selection" examine whether a focused *argument* for a certain *predicate type* is erroneous or not. Figure 5 shows the pattern for "Prepositional attachment," which col-

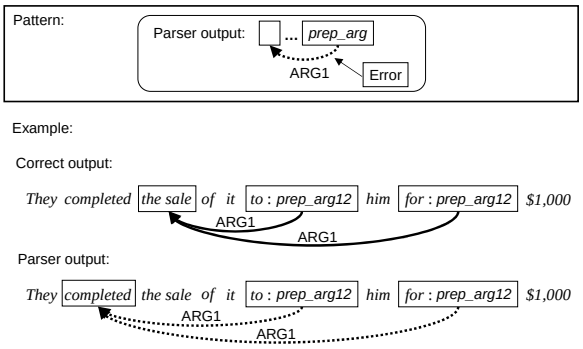


Figure 5: Pattern for "Prepositional attachment"

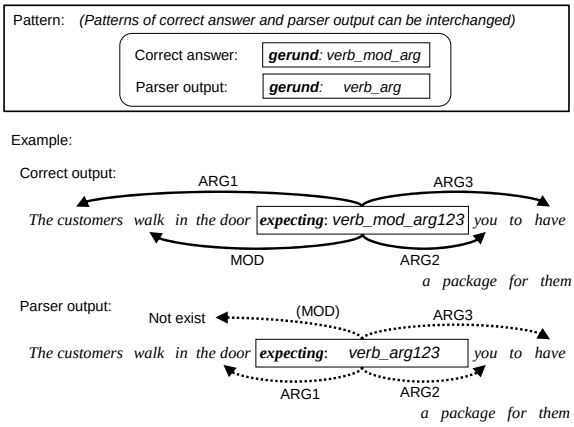


Figure 6: Pattern for "Gerund acts as modifier or not"

lects wrong ARG1 for *predicate type* "*prep_arg*". From the sentence in the figure, we can obtain two errors for "Prepositional attachment" around prepositions "*to*" and "*for*." On the other hand, each "Predicate type selection" pattern collects errors around a word whose *predicate type* is erroneous. Figure 6 shows the pattern for "Gerund acts as modifier or not," which collects errors around gerunds whose *predicate types* are erroneous. From the example sentence in the figure, we can obtain an erroneous predicate type for "*expecting*" and collect errors around it for "Gerund acts as modifier or not."

We can implement more structural errors than simple *argument* or *predicate type* selections. Figures 7 and 8 show the patterns for "To-infinitive for modifier/argument of verb" and "Subject for passive sentence or not" respectively. The pattern for the latter phenomenon collects errors on recognitions of prepositional phrases which behave as subjects for passive expressions. The pattern collects errors not only around prepositions but also around the verbs which take the prepositional phrases as a subject.

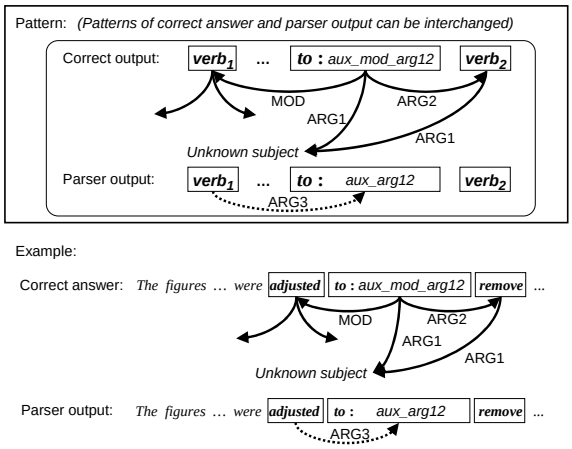


Figure 7: Pattern for "To-infinitive for modifier/argument of verb"

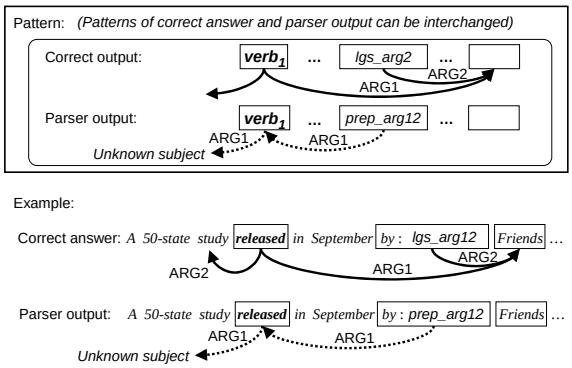


Figure 8: Pattern for "Subject for passive sentence or not"

Since these patterns are based on linguistic knowledge given by a human, the process could provide a relatively precise analysis with a lower cost than a totally manual analysis.

3.2 Empirical approach

Our empirical approach, on the other hand, briefly traces the parsing process which results in each of the target errors. We collect co-occurring errors as strongly relevant ones, and then extract dependencies among the obtained groups. Parsing errors could originate from wrong processing at certain stages in the parsing, and errors with a common origin would by necessity appear together. We reparse a target sentence under the condition where a certain error is forcibly corrected and then collect errors which are corrected together as the "*relative*" ones. An error group where all errors are *relative* to each other can be regarded as a "co-occurring error group." Errors in the same co-

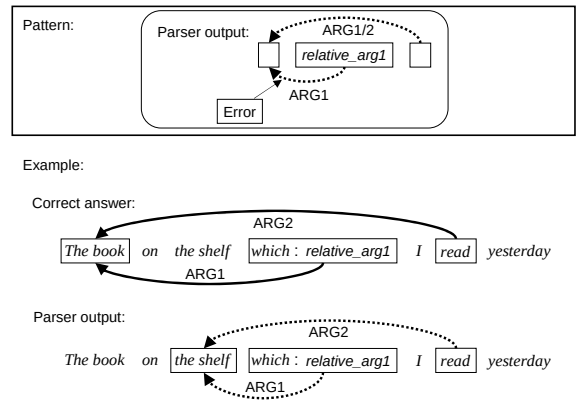


Figure 9: Pattern for "Relative clause attachment"

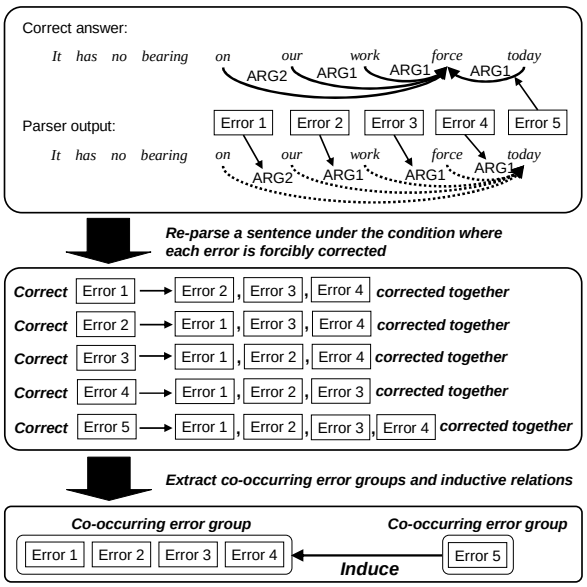


Figure 10: An image of our empirical approach

occurring error group are expected to participate in the same phenomenon. Dependencies among errors are then expected to be summarized with inductions among co-occurring error groups.

Figure 10 shows an image of this approach. In this example, "*today*" should modify noun phrase "*our work force*" while the parser decided that "*today*" was also in the noun phrase. As a result, there are five errors: three wrong outputs for "ARG2" of "*on*" (Error 1) and "ARG1" of "*our*" (Error 2) and "*work*" (Error 3), excess relation "ARG1" of "*force*" (Error 4), and missing relation "ARG1" for "*today*" (Error 5). By correcting each of the errors 1, 2, 3 and 4, all of these errors are corrected together, and therefore classified into the same co-occurring error group. Although error 5 cannot participate in the group, correcting error 5 can correct all of the errors in the group, and therefore an

Error types	# of Errors	# of Patterns
· **Analyzed**	2,078	1,671
[Argument selection]		
Prepositional attachment	579	579
Adjunction attachment	261	261
Conjunction attachment	43	40
Head selection for noun phrase	30	30
Coordination	202	184
[Predicate type selection]		
Preposition/Adjunction	108	54
Gerund acts as modifier or not	84	31
Coordination/conjunction	54	27
# of arguments for preposition	51	17
Adjunction/adjunctive noun	13	13
[More structural errors]		
To-infinitive for modifier/argument of verb	120	22
Subject for passive sentence or not	8	3
[Others]		
Comma	444	372
Relative clause attachment	102	38
· **Unanalyzed**	2,631	–
Total	4,709	–

Table 3: Errors extracted with descriptive analysis

inductive relation is given from error 5 to the co-occurring error group. We can then finally obtain the inductive relations as shown at the bottom of Figure 10. This approach can trace the actual behavior of the parser precisely, and can therefore capture underlying dependencies which cannot be found only by observing error outputs.

4 Experiments

We applied our approaches to parsing errors given by the HPSG parser *Enju*, which was trained on the Penn Treebank (Marcus et al., 1994) section 2-21. We first examined each approach, and then explored the combination of the approaches.

4.1 Evaluation of descriptive approach

We examined our descriptive approach. We first parsed sentences in the Penn Treebank section 22 with *Enju*, and then observed the errors. Based on the observation, we next described the patterns as shown in Section 3. After that, we parsed section 0 and then applied the patterns to the errors.

Table 3 summarizes the extracted errors. As the table shows, with the 14 error patterns, we successfully matched 1,671 locations in error outputs and covered 2,078 of 4,709 errors, which comprised of more than 40% of the total errors. This was the first step of the application of our approach, and in the future work we would like to

Evaluated sentences (erroneous)	1,811 (1,009)
Errors (Correctable)	4,709 (3,085)
Co-occurring errors	1,978
Extracted inductive relations	501
F-score (LP/LR)	90.69 (90.78/93.59)

Table 4: Summary of our empirical approach

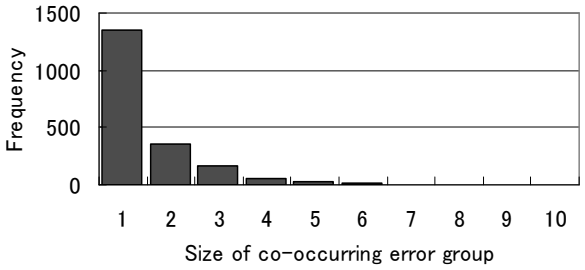


Figure 11: Frequency of each size of co-occurring error group

add more patterns for capturing more phenomena.

When we focused on individual patterns, we could observe that the simple error phenomena such as the attachments were dominant. The first reason for this would be that such phenomena were among minimal linguistic events. This would make the phenomena components of other more complex ones. The second reason for the dominance would be that the patterns for these error phenomena were easy to implement only with argument inconsistencies, and only one or a few patterns could cover every probable error. Among these dominant error types, the number of prepositional attachments was outstanding. The error types which required matching with predicate types were fewer than the attachment errors since the limited patterns on the predicate types would narrow the possible linguistic behavior of the candidate words. When we focus on more structural errors, the table shows that the rates of the participant errors to matched locations were much larger than those for simpler pattern errors. Once our patterns matches, they could collect many errors at the same time.

4.2 Evaluation of empirical approach

Next, we applied our empirical approach in the same settings as in the previous section. We first parsed sentences in section 0 and then applied our approach to the obtained errors. In the experiments, some errors could not be forcibly corrected by our approach. The parser "*cut off*" less probable parse substructures before giving the predicate

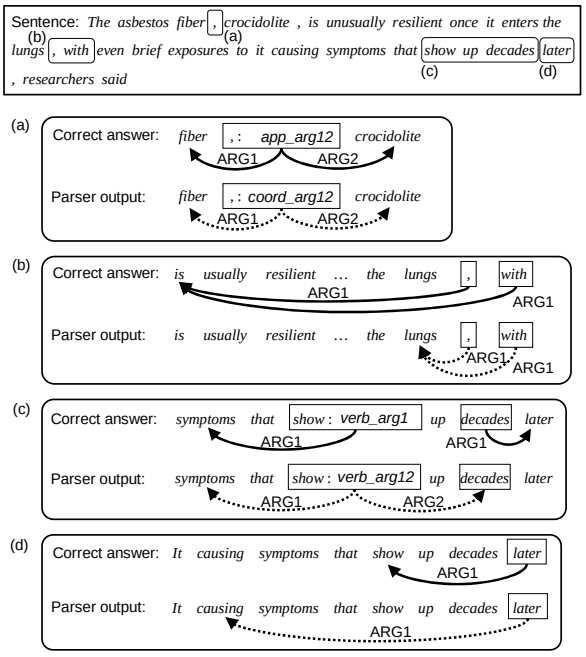


Figure 12: Obtained co-occurring error groups

argument relation for reducing the cost of parsing. In this research, we ignored the errors which were subject to such "cut off" as "*uncorrectable*" ones, and focused only on the remaining "*correctable*" errors. In our future work, we would like to consider the "*uncorrectable*" errors.

Table 4 shows the summary of the analysis with our approach. *Enju* gave 4,709 errors for section 0. Among these errors, the *correctable* errors were 3,085, and from these errors, we successfully obtained 1,978 co-occurring error groups and 501 inductive relations. Figure 11 shows the frequency for each size of co-occurring groups. About a half of the groups contains only single errors, which would indicate that the errors could have only one-way inductive relations with other errors. The rest of this section explores examples of the obtained co-occurring error groups and inductive relations.

Figure 12 shows an example of the extracted co-occurring error groups. For the sentence shown at the top of the figure, *Enju* gave seven errors. By introducing our empirical approach, these errors were definitely classified into four co-occurring error groups (a) to (d), and there were no inductive relations detected among them. Group (a) contains two errors on the comma's local behavior as apposition or coordination. Group (b) contains the errors on the words which gave almost the same attachment behaviors. Group (c) contains the errors on whether the verb "*show*" took "*decades*"

Error types	# of correctable errors	# of independent errors	Correction effect (errors)
[Argument selection]			
Prepositional attachment	531	397	766
Adjunction attachment	196	111	352
Conjunction attachment	33	12	79
Head selection for noun phrase	22	0	84
Coordination	146	62	323
[Predicate type selection]			
Preposition/Adjunction	72	30	114
Gerund acts as modifier or not	39	18	62
Coordination/conjunction	36	16	61
# of arguments for preposition	24	23	26
Adjunction/adjunctive noun	8	6	10
[More structural errors]			
To-infinitive for modifier/argument of verb	75	27	87
Subject for passive sentence or not	8	3	9
[Others]			
Comma	372	147	723
Relative clause attachment	84	27	119
Total	1,646	979	–

Table 5: Induction relations between errors for each linguistic phenomenon and other errors

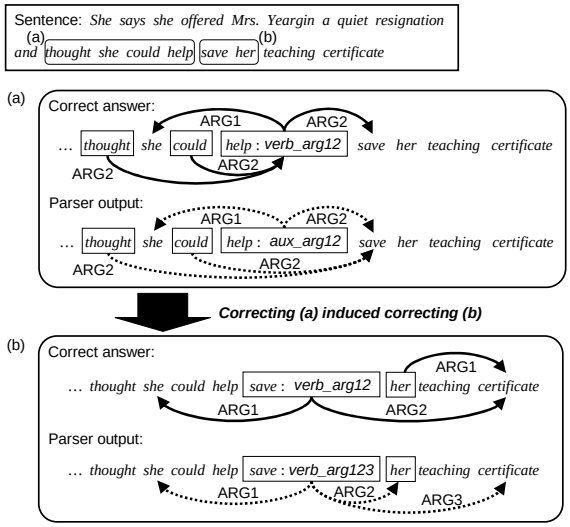


Figure 13: Inductive relation between obtained co-occurring error groups

as its object or not. Group (d) contains an error on the attachment of the adverb "*later*". Regardless of the overlap of the regions in the sentence for (c) and (d), our approach successfully classified the errors into the two independent groups. With our approach, it would be empirically shown that the errors in each group actually co-occurred and the group was independent. This would enable us to concentrate on each of the co-occurring error groups without paying attention to the influences from the errors in other groups.

Figure 13 shows another example of the analysis with our empirical approach. In this case, 8 errors for a sentence were classified into two co-occurring error groups (a) and (b), and our approach showed that correction in group (a) resulted in correcting group (b) together. The errors in group (a) were on whether "*help*" behaved as an auxiliary or pure verbal role. The errors in group (b) were on whether "*save*" took only one object "*her teaching certificate*," or two objects "*her*" and "*teaching certificate*." Between group (a) and (b), no "structural" conflict could be found when correcting only each of the groups. We could then guess that the inductive relation between these two groups was implicitly given by the disambiguation model of the parser. By dividing the errors into minimum units and clarifying the effects of correcting a target error, error analysis with our empirical approach could suggest some policy for parser improvements.

4.3 Combination of two approaches

On the basis of the experiments shown in the previous sections, we would like to explore possibilities for obtaining a more detailed analysis by combining the two approaches.

4.3.1 Interactions between a target linguistic phenomenon and other errors

Our descriptive approach could classify the parsing errors according to the linguistic phenomena they participated in. We then attempt to reveal how such classified errors interacted with other errors from the viewpoints of our empirical approach. In order to enable the analysis by our empirical approach, we focused only on the *correctable* errors.

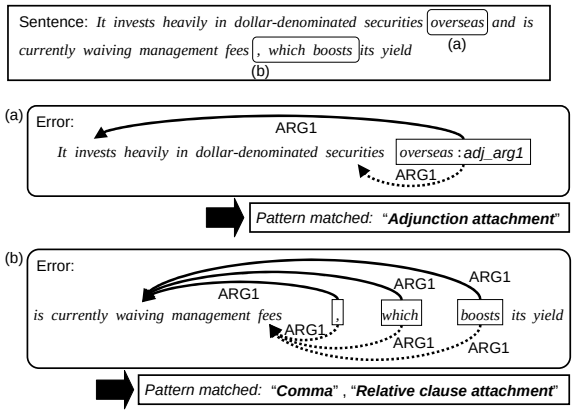


Figure 14: Combination of results given by descriptive and empirical approaches (1)

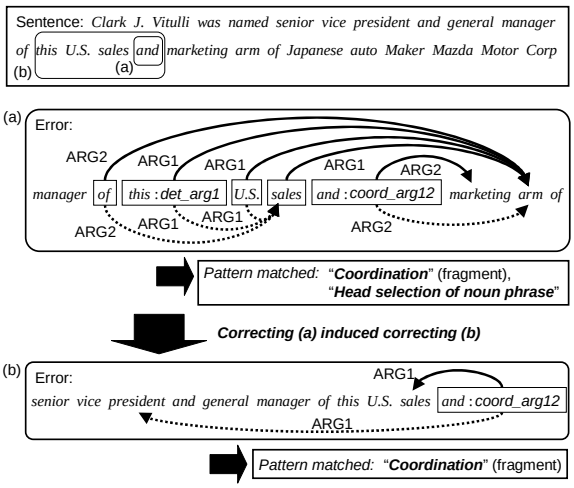


Figure 15: Combination of results given by descriptive and empirical approaches (2)

Table 5 reports the degree to which the classified errors were related to other individual errors. The leftmost numbers show the numbers of *correctable* errors, which were the focused errors in the experiments. The central numbers show the numbers of "*independent*" errors, that is, the errors which could be corrected only by correcting themselves. The rightmost numbers show "*correction effects*," that is, the number of errors which would consequently be corrected if all of the errors for the focused phenomena were forcibly corrected.

"*Independent*" errors are obtained by collecting error phenomena groups which consist of unions of co-occurring error groups and each error in which is not induced by other errors. Figure 14 shows an example of "*independent*" errors. For the sentence at the top of the figure, the parser had four errors on ARG1 of "*overseas*," the comma, "*which*" and "*boosts*." Our empirical approach then classified these errors into two co-occurring error groups (a) and (b), and there was no inductive relation between the groups. Our descriptive approach, on the other hand, matched all of the errors with the patterns for "Adjunction attachment," "Comma" and "Relative clause attachment." Since the error for the "Adjunction attachment" equals to a co-occurring group (a) and is not induced by other errors, the error is "*independent*."

Table 5 shows that, for "Prepositional attachment", "Adjunction attachments," "# of arguments for preposition" and "Adjunction/adjunctive noun," more than half of the errors for the focused phenomena are "*independent*." Containing many "*independent*" errors would mean that the parser should handle these phenomena further more intensively as an independent event.

The "*correction effect*" for a focused linguistic phenomenon can be obtained by counting errors in the union of the *correctable* error set for the phenomenon and the error sets which were induced by the individual errors in the set. We would show an example of *correction effect* in Figure 15. In the figure, the parser had six errors for the sentence at the top: three false outputs for ARG1 of "*and*," "*this*" and "*U.S.*," two false outputs for ARG2 of "*of*" and "*and*," and missing output for ARG1 of "*sales*." Our empirical approach classified these errors into two co-occurring error groups (a) and (b), and extracted an inductive relation from (a) to (b). Our descriptive approach, on the other hand, matched two errors on "*and*" with pattern "Coordination" and one error on "*this*" with "Head selection for noun phrase." When we focus on the error for "Head selection of noun phrase" in co-occurring group (a), the correction of the error induced the rest of the errors in (a), and further induced the error in (b) according to the inductive relation from (a) to (b). Therefore, a "*correction effect*" for the error results in six errors.

Table 5 shows that, for "Conjunction attachment," "Head selection for noun phrase" and "Coordination," each "*correction effect*" results in more than twice the forcibly corrected errors. Improving the parser so that it can resolve such *high-correction-effect* erroneous phenomena may additionally improve the parsing performances to a great extent. On the other hand, "Head selection for noun phrase" contains no "*independent*" error, and therefore could not be handled independently of other erroneous phenomena at all. Consider-

ing the effects from outer events might make the treatment of "Head selection for noun phrase" a more complicated process than other phenomena, regardless of its high "*correction effect*."

Table 5 would thus suggest which phenomenon we should resolve preferentially from the three points of view: the number of errors, the number of "*independent*" errors and its "*correction effect*." Considering these points, "Prepositional attachment" seems most preferable for handling first.

4.3.2 Possibilities for further analysis

Since the errors for the phenomenon were systematically collected with our descriptive approach, we can work on further focused error analyses which would answer such questions as "Which preposition causes most errors in attachments?", "Which pair of a correct answer and an erroneous output for predicate argument relations can occur most frequently?", and so on. Our descriptive approach would enable us to thoroughly obtain such analyses with more closely-defined patterns. In addition, our empirical approach would clarify the influences of the obtained error properties on the parser's behaviors. The results of the focused analyses might reasonably lead us to the features that can be captured as parameters for model training, or policies for re-ranking the parse candidates.

The combination of our approaches would give us interesting clues for planning effective strategies for improving the parser. Our challenges for combining the two approaches are now in the preliminary stage and there would be many possibilities for further detailed analysis.

5 Related work

Although there have been many researches which analyzed errors on their own systems in the part of the experiments, there have been few researches which focused mainly on error analysis itself.

In the field of parsing, McDonald and Nivre (2007) compared parsing errors between graph-based and transition-based parsers. They observed the accuracy transitions from various points of view, and the obtained statistical data suggested that error propagation seemed to occur in the graph structures of parsing outputs. Our research proceeded for one step in this point, and attempted to reveal the way of the propagations. In examining the combination of the two types of parsing, McDonald and Nivre (2007) utilized similar approaches to our empirical analysis. They allowed a parser to give only structures given by the parsers. They implemented the ideas for evaluating the parser's potentials whereas we implemented the ideas for observing error propagations.

Dredze et al. (2007) showed the possibility that many parsing errors in the domain adaptation tasks came from inconsistencies between annotation manners of training resources. Such findings would further suggest that, comparing given errors without considering the inconsistencies could lead to the misunderstanding of what occurs in domain transitions. The summarized error dependencies given by our approaches would be useful clues for extracting such domain-dependent error phenomena.

Giménez and Màrquez (2008) proposed an automatic error analysis approach in machine translation (MT) technologies. They were developing a metric set which could capture features in MT outputs at different linguistic levels with different levels of granularity. As we considered the parsing systems, they explored the way to resolve costly and non-rewarding error analysis in the MT field. One of their objectives was to enable researchers to easily access detailed linguistic reports on their systems and to concentrate only on analyses for the system improvements. From this point of view, our research might provide an introduction into such rewarding analysis in parsing.

6 Conclusions

We proposed empirical and descriptive approaches to extracting dependencies among parsing errors. In the experiments, with each of our approaches, we successfully obtained relevant errors. Moreover, the possibility was shown that the combination of our approaches would give a more detailed error analysis which would bring us useful clues for parser improvements.

In our future work, we will improve the performance of our approaches by adding more patterns for the descriptive approach and by handling *uncorrectable* errors for the empirical approach. With the obtained robust information, we will explore rewarding ways for parser improvements.

Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan).

References

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, João V. Graça, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *Proceedings of the CoNLL Shared Task Session of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1051–1055.

Jesús Giménez and Lluís Màrquez. 2008. Towards heterogeneous automatic MT error analysis. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, pages 1894–1901.

Ronald M. Kaplan and Joan Bresnan. 1995. Lexical-functional grammar: A formal system for grammatical representation. *Formal Issues in Lexical-Functional Grammar*, pages 29–130.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert Macintyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of ARPA Human Language Technology Workshop*.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 83–90.

Takashi Ninomiya, Takuya Matsuzaki, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Extremely lexicalized models for accurate and fast HPSG parsing. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 155–163.

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

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

Large-Scale Verb Entailment Acquisition from the Web

Chikara Hashimoto[*] **Kentaro Torisawa**[†] **Kow Kuroda**[‡]
Stijn De Saeger[§] **Masaki Murata**[¶] **Jun'ichi Kazama**[‖]
National Institute of Information and Communications Technology
Sorakugun, Kyoto, 619-0289, JAPAN
{[*] ch,[†] torisawa,[‡] kuroda,[§] stijn,[¶] murata,[‖] kazama}@nict.go.jp

Abstract

Textual entailment recognition plays a fundamental role in tasks that require in-depth natural language understanding. In order to use entailment recognition technologies for real-world applications, a large-scale entailment knowledge base is indispensable. This paper proposes a conditional probability based directional similarity measure to acquire verb entailment pairs on a large scale. We targeted 52,562 verb types that were derived from 10^8 Japanese Web documents, without regard for whether they were used in daily life or only in specific fields. In an evaluation of the top 20,000 verb entailment pairs acquired by previous methods and ours, we found that our similarity measure outperformed the previous ones. Our method also worked well for the top 100,000 results.

1 Introduction

We all know that if you snored, you must have been sleeping, that if you are divorced, you must have been married, and that if you won a lawsuit, you must have sued somebody. These relationships between events where one is the logical consequence of the other are called entailment. Such knowledge plays a fundamental role in tasks that require in-depth natural language understanding, e.g., answering questions and using natural language interfaces.

This paper proposes a novel method for verb entailment acquisition. Using a Japanese Web corpus (Kawahara and Kurohashi, 2006a) derived from 10^8 Japanese Web documents, we automatically acquired such verb pairs as *snore* → *sleep* and *divorce* → *marry*, where entailment holds between the verbs in the pair.[1] Our definition of "entailment" is the same as that in WordNet3.0; v_1 entails v_2 if v_1 cannot be done unless v_2 is, or has been, done.[2]

Our method follows the distributional similarity hypothesis, i.e., words that occur in the same context tend to have similar meanings. Just as in the methods of Lin and Pantel (2001) and Szpektor and Dagan (2008), we regard the arguments of verbs as the context in the hypothesis. However, unlike the previous methods, ours is based on conditional probability and is augmented with a simple trick that improves the accuracy of verb entailment acquisition. In an evaluation of the top 20,000 verb entailment pairs acquired by the previous methods and ours, we found that our similarity measure outperformed the previous ones. Our method also worked well for the top 100,000 results,

Since the scope of Natural Language Processing (NLP) has advanced from a formal writing style to a colloquial style and from restricted to open domains, it is necessary for the language resources for NLP, including verb entailment knowledge bases, to cover a broad range of expressions, regardless of whether they are used in daily life or only in specific fields that are highly technical. As we will discuss later, our method can acquire, with reasonable accuracy, verb entailment pairs that deal not only with common and familiar verbs but also with technical and unfamiliar ones like *podcast* → *download* and *jibe* → *sail*.

Note that previous researches on entailment acquisition focused on templates with variables or word-lattices (Lin and Pantel, 2001; Szpektor and Dagan, 2008; Barzilay and Lee, 2003; Shinyama

[1] Verb entailment pairs are described as $v_1 \rightarrow v_2$ (v_1 is the entailing verb and v_2 is the entailed one) henceforth.

[2] WordNet3.0 provides entailment relationships between synsets like *divorce, split up* → *marry, get married, wed, conjoin, hook up with, get hitched with, espouse*.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1172–1181,
Singapore, 6-7 August 2009.

et al., 2002). Certainly these templates or word lattices are more useful in such NLP applications as Q&A than simple entailment relations between verbs. However, our contention is that entailment certainly holds for some verb pairs (like *snore* → *sleep*) by themselves, and that such pairs constitute the core of a future entailment rule database. Although we focused on verb entailment, our method can also acquire template-level entailment pairs with a reasonable accuracy.

The rest of this paper is organized as follows. In §2, related works are described. §3 presents our proposed method. After this, an evaluation of our method and the existing methods is presented in Section 4. Finally, we conclude the paper in §5.

2 Related Work

Previous studies on entailment, inference rules, and paraphrase acquisition are roughly classified into those that require comparable corpora (Shinyama et al., 2002; Barzilay and Lee, 2003; Ibrahim et al., 2003) and those that do not (Lin and Pantel, 2001; Weeds and Weir, 2003; Geffet and Dagan, 2005; Pekar, 2006; Bhagat et al., 2007; Szpektor and Dagan, 2008).

Shinyama et al. (2002) regarded newspaper articles that describe the same event as a pool of paraphrases, and acquired them by exploiting named entity recognition. They assumed that named entities are preserved across paraphrases, and that text fragments in the articles that share several comparable named entities should be paraphrases. Barzilay and Lee (2003) also used newspaper articles on the same event as comparable corpora to acquire paraphrases. They induced paraphrasing patterns by sentence clustering. Ibrahim et al. (2003) relied on multiple English translations of foreign novels and sentence alignment to acquire paraphrases. We decided not to take this approach since using comparable corpora limits the scale of the acquired paraphrases or entailment knowledge bases. Although obtaining comparable corpora has been simplified by the recent explosion of the Web, the availability of plain texts is incomparably better.

Entailment acquisition methods that do not require comparable corpora are mostly based on the distributional similarity hypothesis and use plain texts with a syntactic parser. Basically, they parse texts to obtain pairs of predicate phrases and their arguments, which are regarded as features of the predicates with appropriately assigned weights. Lin and Pantel (2001) proposed a paraphrase acquisition method (non-directional similarity measure) called DIRT which acquires pairs of binary-templates (predicate phrases with two argument slots) that are paraphrases of each other. DIRT employs the following similarity measure proposed by Lin (1998):

$$Lin(l,r) = \frac{\sum_{f \in F_l \cap F_r} [w_l(f) + w_r(f)]}{\sum_{f \in F_l} w_l(f) + \sum_{f \in F_r} w_r(f)}$$

where l and r are the corresponding slots of two binary templates, F_s is s's feature vector (argument nouns), and $w_s(f)$ is the weight of $f \in F_s$ (PMI between s and f). The intuition behind this is that the more nouns two templates share, the more semantically similar they are. Since we acquire verb entailment pairs based on unary templates (Szpektor and Dagan, 2008) we used the Lin formula to acquire unary templates directly rather than using the DIRT formula, which is the arithmetic-geometric mean of Lin's similarities for two slots in a binary template.

Bhagat et al. (2007) developed an algorithm called LEDIR for learning the directionality of non-directional inference rules like those produced by DIRT. LEDIR implements a Directionality Hypothesis: when two binary semantic relations tend to occur in similar contexts and the first one occurs in significantly more contexts than the second, then the second most likely implies the first and not vice versa.

Weeds and Weir (2003) proposed a general framework for distributional similarity that mainly consists of the notions of what they call Precision (defined below) and Recall:

$$Precision(l,r) = \frac{\sum_{f \in F_l \cap F_r} w_l(f)}{\sum_{f \in F_l} w_l(f)}$$

where l and r are the targets of a similarity measurement, F_s is s's feature vector, and $w_s(f)$ is the weight of $f \in F_s$. The best performing weight is PMI. Precision is a directional similarity measure that examines the coverage of l's features by those of r's, with more coverage indicating more similarity.

Szpektor and Dagan (2008) proposed a directional similarity measure called BInc (Balanced-Inclusion) that consists of Lin and Precision, as

$$BInc(l,r) = \sqrt{Lin(l,r) \times Precision(l,r)}$$

where l and r are the target templates. For weighting features, they used PMI. Szpektor and Dagan (2008) also proposed a unary template, which is defined as a template consisting of one argument slot and one predicate phrase. For example, *X take a nap* → *X sleep* is an entailment pair consisting of two unary templates. Note that the slot *X* must be shared between templates. Though most of the previous entailment acquisition studies focused on binary templates, unary templates have an obvious advantage over binary ones; they can handle intransitive predicate phrases and those that have omitted arguments. The Japanese language, which we deal with here, often omits arguments, and thus the advantage of unary templates is obvious.

As shown in §4, our method outperforms Lin, Precision, and BInc in accuracy.

Szpector et al. (2004) addressed broad coverage entailment acquisition. But their method requires an existing lexicon to start, while ours does not.

Apart from the dichotomy of the comparable corpora and the distributional similarity approaches, Torisawa (2006) exploited the structure of Japanese coordinated sentences to acquire verb entailment pairs. Pekar (2006) used the local structure of coherent text by identifying related clauses within a local discourse. Zanzotto et al. (2006) exploited agentive nouns. For example, they acquired *win* → *play* from "*the player wins*."

Geffet and Dagan (2005) proposed the Distributional Inclusion Hypotheses, which claimed that if a word *v* entails another word *w*, then all the characteristic features of *v* are expected to appear with *w*, and vice versa. They applied this to noun entailment pair acquisition, rather than verb pairs.

3 Proposed Method

This section presents our method of verb entailment acquisition. First, the basics of Japanese are described. Then, we present the directional similarity measure that we developed in §3.2. §3.3 describes the structure and acquisition of the web-based data from which entailment pairs are derived. Finally, we show how we acquire verb entailment pairs using our proposed similarity measure and the web-based data in §3.4.

3.1 Basics of Japanese

Japanese explicitly marks arguments including the subject and object by postpositions, and is a head-final language. Thus, a verb phrase consisting of an object *hon* (book) and a verb *yomu* (read), for example, is expressed as *hon-wo yomu* (book-ACC read) "read a book" with the accusative postposition *wo* marking the object.[3] Accordingly, we refer to a unary template as $\langle p, v \rangle$ hereafter, with p and v referring to the *p*ostposition and a *v*erb. Also, we abbreviate a template-level entailment $\langle p_l, v_l \rangle \rightarrow \langle p_r, v_r \rangle$ as $l \rightarrow r$ for simplicity. We define a unary template as a template consisting of one argument slot and one predicate, following Szpektor and Dagan (2008).

3.2 Directional Similarity Measure based on Conditional Probability

The directional similarity measure that we developed and called $Score$ is defined as follows:

$$Score(l, r) = Score_{base}(l, r) \times Score_{trick}(l, r)$$

where l and r are unary templates, and $Score$ indicates the probability of $l \rightarrow r$. $Score_{base}$, which is the base of $Score$, is defined as follows:

$$Score_{base}(l, r) = \sum_{f \in F_l \cap F_r} P(r|f)P(f|l)$$

where F_s is s's feature vector (nouns including compounds). The intention behind the definition of $Score_{base}$ is to *emulate* the conditional probability $P(v_r|v_l)$[4] in a distributional similarity style function. Note that $P(v_r|v_l)$ should be 1 when entailment $v_l \rightarrow v_r$ holds (i.e., v_r is observed whenever v_l is observed) and we have reliable probability values. Then, if we can directly estimate $P(v_r|v_l)$, it is reasonable to assume $v_l \rightarrow v_r$ if $P(v_r|v_l)$ is large enough. However, we cannot estimate $P(v_r|v_l)$ directly since it is unlikely that we will observe the verbs v_r and v_l at the same time. (People do not usually repeat v_r and v_l in the same document to avoid redundancy.) Thus, instead of a direct estimation, we *substitute* $Score_{base}(l, r)$ as defined above. In other words, we assume $P(v_r|v_l) \approx P(r|l) \approx \Sigma_{f \in F_l \cap F_r} P(f|l)P(r|f)$.

Actually, $Score_{base}$ originally had another motivation, inspired by Torisawa (2005), for which no postposition but the instrumental postposition *de* was relevant. In this discussion, all of the nouns (fs) that are marked by the instrumental postposition are seen as "tools," and $P(f|l)$ is interpreted

[3] ACC represents an accusative postposition in Japanese. Likewise, NOM, DAT, INS, and TOP are the symbols for the nominative, dative, instrumental, and topic postpositions.

[4] Remember that v_l and v_r are the verbs of unary templates l and r.

as a measure of how typically the tool f is used to perform the action denoted by (the v_l of) l; if $P(f|l)$ is large enough, f is a typical tool used in l. On the other hand, $P(r|f)$ indicates the probability of (the v_r of) r being the purpose for using the tool f. See (1) for an example.

(1) *konro-de* *chouri-suru*
cooking.stove-INS cook
'cook (something) using a cooking stove.'

The purpose of using a *cooking stove* is to *cook*. Torisawa (2005) has pointed out that when r expresses the purpose of using a tool f, $P(r|f)$ tends to be large. This predicts that $P(r|$*cooking stove*$)$ is large, where r is $\langle de, cook \rangle$.

According to this observation, if f is a single purpose tool and $P(f|l)$, the probability of f being the tool by which l is performed, and $P(r|f)$, the probability of r being the purpose of using the tool f, are large enough, then the typical performance of the action v_l should contain some actions that can be described by v_r, i.e., the purpose of using f. Moreover, if all the typical tools (fs) used in v_l are also used for v_r, most performances of the action v_l should contain a part described by the action v_r. In summary, this means that when $\Sigma_{f \in F_l \cap F_r} P(r|f)P(f|l)$, $Score_{base}$, has a large value, we can expect $v_l \rightarrow v_r$.

For example, let v_l be *deep-fry* and v_r be *cook*. Note that $v_l \rightarrow v_r$ holds for this example. There are many tools that are used for deep-frying, such as *cooking stove*, *pot*, or *pan*. This means that $P($*cooking stove*$|l)$, $P($*pot*$|l)$, or $P($*pan*$|l)$ are large. On the other hand, the purpose of using all of these tools is *cooking*, based on common sense. Thus, probabilities such as $P(r|$*cooking stove*$)$ and $P(r|$*pan*$)$ should have large values. Accordingly, $\Sigma_{f \in F_l \cap F_r} P(f|l)P(r|f)$, $Score_{base}$, should be relatively large for *deep-fry* $\rightarrow$ *cook*,

Actually, we defined $Score_{base}$ based on the above assumption However, through a series of preliminary experiments, we found that the same score could be applied without losing the precision to the other postpositions. Thus, we generalized the framework so that it could deal with most postpositions, namely *ga* (NOM), *wo* (ACC), *ni* (DAT), *de* (INS), and *wa* (TOP). Note that this is a variation of the distributional inclusion hypothesis (Geffet and Dagan, 2005), but that we do not use mutual information as in previous works, based on the hypothesis discussed above. Actually, as shown in §4, our conditional probability based method outperformed the mutual information based metrics in our experiments.

On the other hand, $Score_{trick}$ implements another assumption that if only one feature contributes to $Score_{base}$ and the contribution of the other nouns is negligible, if any, the similarity is unreliable. Accordingly, for $Score_{trick}$, we uniformly ignore the contribution of the most dominant feature from the similarity measurement.

$$Score_{trick}(l, r) = Score_{base}(l, r) - \max_{f \in F_l \cap F_r} P(r|f)P(f|l)$$

As shown in §4, this trick actually improved the entailment acquisition accuracy.

We used maximum likelihood estimation to obtain $P(r|f)$ and $P(f|l)$ in the above discussion.

Bannard and Callison-Burch (2005) and Fujita and Sato (2008) also proposed directional similarity measures based on conditional probability, which are very similar to $Score_{base}$, although either their method's prerequisites or the targets of the similarity measurements were different from ours. The method of Bannard and Callison-Burch (2005) requires bilingual parallel corpora, and uses the translations of expressions as its feature. Fujita and Sato (2008) dealt with productive predicate phrases, while our target is non-productive lexical units, i.e., verbs. Thus, this is the first attempt to apply a conditional probability based similarity measure to verb entailment acquisition. In addition, the trick implemented in $Score_{trick}$ is novel.

3.3 Preparing Template-Feature Tuples

Our method starts from a dataset called template-feature tuples, which was derived from the Web in the following way: **1)** Parse the Japanese Web corpus (Kawahara and Kurohashi, 2006a) derived from 10^8 Japanese Web documents with Japanese dependency parser KNP (Kawahara and Kurohashi, 2006b). **2)** Extract triples $\langle n, p, v \rangle$ consisting of nouns (n), postpositions (p), and verbs (v), where an n marked by a p depends on a v from the parsed Web text. **3)** From the triple database, construct template-feature tuples $\langle n, \langle p, v \rangle \rangle$ by regarding $\langle p, v \rangle$ as a unary template and n as one of its features. **4)** Convert the verbs into their canonical forms as defined by KNP. **5)** Filter out tuples that fall into one of the following categories: 5-1) $Freq(\langle p, v \rangle) < 20$. 5-2) Its verb is passivized,

causativized, or negated. 5-3) Its verb is semantically vague like *be*, *do*, or *become*. 5-4) Its postposition is something other than *ga* (NOM), *wo* (ACC), *ni* (DAT), *de* (INS), or *wa* (TOP).

The resulting unary template-feature tuples included 127,808 kinds of templates that consisted of 52,562 verb types and five kinds of postpositions. The verbs included compound words like *bosi-kansen-suru* (mother.to.child-infection-do) "infect from mothers to infants."

3.4 Acquiring Entailment Pairs

We acquired verb entailment pairs using the following procedure: **i)** From the template-feature tuples mentioned in §3.3, acquire unary template pairs that exhibit an entailment relation between them using the directional similarity measure in §3.2. **ii)** Convert the acquired unary templates $\langle p, v \rangle$ into naked verbs v by stripping the postpositions p. **iii)** Remove the duplicated verb pairs resulting from stripping ps. To be precise, when we removed the duplicated pairs, we left the highest ranked one. **iv)** Retrieve N-best verb pairs as the final output from the result of iii). That is, we first acquired unary template pairs and then transformed them into verb pairs.

Although this paper focuses on verb entailment acquisition, we also evaluated the accuracy of template-level entailment acquisition, in order to show that our similarity measure works well, not only for verb entailment acquisition, but also for template entailment acquisition (See §4.4). we created two kinds of unary templates: the "Scoring Slots" template and the "Nom(inative) Slots" template. The first is simply the result of the procedure **i)**; all of the templates have slots that are used for similarity scoring. The second one was obtained in the following way: **1)** Only templates whose p is not a nominative are sampled from the result of the procedure **i)**. **2)** Their ps are all changed to a nominative. Templates of the second kind are used to show that the corresponding slots between templates (nominative, in this case) that are not used for similarity scoring can be incorporated to resulting template-level entailment pairs if the scoring function really captures the semantic similarity between templates.

Note that, for unary template entailment pairs like (2) to be well-formed, the two unary slots (*X-wo*) between templates must share the same noun as the index i indicates. This is relevant in §4.4.

(2) *X_i-wo musaborikuu* → *X_i-wo taberu*
X_i-ACC gobble X_i-ACC eat

4 Evaluation

We compare the accuracy of our method with that of the alternative methods in §4.1. §4.2 shows the effectiveness of the trick. We examine the entailment acquisition accuracy for frequent verbs in §4.3, and evaluate the performance of our method when applied to template-level entailment acquisition in §4.4. Finally, by showing the accuracy for verb pairs obtained from the top 100,000 results, we claim that our method provides a good starting point from which a large-scale verb entailment resource can be constructed in §4.5.

For the evaluation, three human annotators (not the authors) checked whether each acquired entailment pair was correct. The average of the three Kappa values for each annotator pair was 0.579 for verb entailment pairs and 0.568 for template entailment pairs, both of which indicate the middling stability of this evaluation annotation.

4.1 Experiment 1: Verb Pairs

We applied $Score$, BInc, Lin, and Precision to the template-feature tuples (§3.3), obtained template entailment pairs, and finally obtained verb entailment pairs by removing the postpositions from the templates as described in §3. As a baseline, we created pairs from randomly chosen verbs.

Since we targeted all of the verbs that appeared on the Web (under the condition of $Freq(\langle p, v \rangle) \geq 20$), the annotators were confronted with technical terms and slang that they did not know. In such cases, they consulted dictionaries (either printed or machine readable ones) and the Web. If they still could not find the meaning of a verb, they labeled the pair containing the unknown verb as incorrect.

We used the accuracy $= \frac{\text{\# of correct pairs}}{\text{\# of acquired pairs}}$ as an evaluation measure. We regarded a pair as correct if it was judged correct by one (Accuracy-1), two (Accuracy-2), or three (Accuracy-3) annotators.

We evaluated 200 entailment pairs sampled from the top 20,000 for each method (# of acquired pairs $= 200$). For fairness, the evaluation samples for each method were shuffled and placed in one file from which the annotators worked. In this way, they were unable to know which entailment pair came from which method.

Note that the verb entailment pairs produced by Lin do not provide the directionality of entailment. Thus, the annotators decided the directionality of these entailment pairs as follows: **i)** Copy 200 original samples and reverse the order of v_1 and v_2. **ii)** Shuffle the 400 Lin samples (the original and reversed samples) with the other ones. **iii)** Evaluate all of the shuffled pairs. Each Lin pair was regarded as correct if either direction was judged correct. In other words, we evaluated the upper bound performance of the LEDIR algorithm.

Table 1 shows the accuracy of the acquired verb entailment pairs for each method. Figure 1

Method	Acc-1	Acc-2	Acc-3
$Score$	**0.770**	**0.660**	**0.460**
BInc	0.450	0.255	0.125
Precision	0.725	0.545	0.385
Lin	0.590	0.370	0.160
Random	0.050	0.010	0.005

Table 1: Accuracy of verb entailment pairs.

shows the accuracy figures for the N-best entailment pairs for each method, with N being 1,000, 2,000, ..., or 20,000. We observed the following points from the results. First, $Score$ outperformed all the other methods. Second, $Score$ and Precision, which are directional similarity measures, worked well, while Lin, which is a symmetric one, performed poorly even though the directionality of its output was determined manually.

Looking at the evaluated samples, $Score$ successfully acquired pairs in which the entailed verbs generalized entailing verbs that were technical terms. (3) shows examples of $Score$'s outputs.

(3) a. *RSS-haisin-suru* → *todokeru*
RSS-feed-do deliver
"feed the RSS data"

b. *middosippu-maunto-suru* → *tumu*
midship-mounting-do mount
"have (engine) midship-mounted"

The errors made by DIRT (4) and BInc (5) included pairs consisting of technical terms.

(4) *kurakkingu-suru*
software.cracking-do
'crack a (security) system'
→ *koutiku-hosyu-suru*
building-maintenance-do
"build and maintain a system"

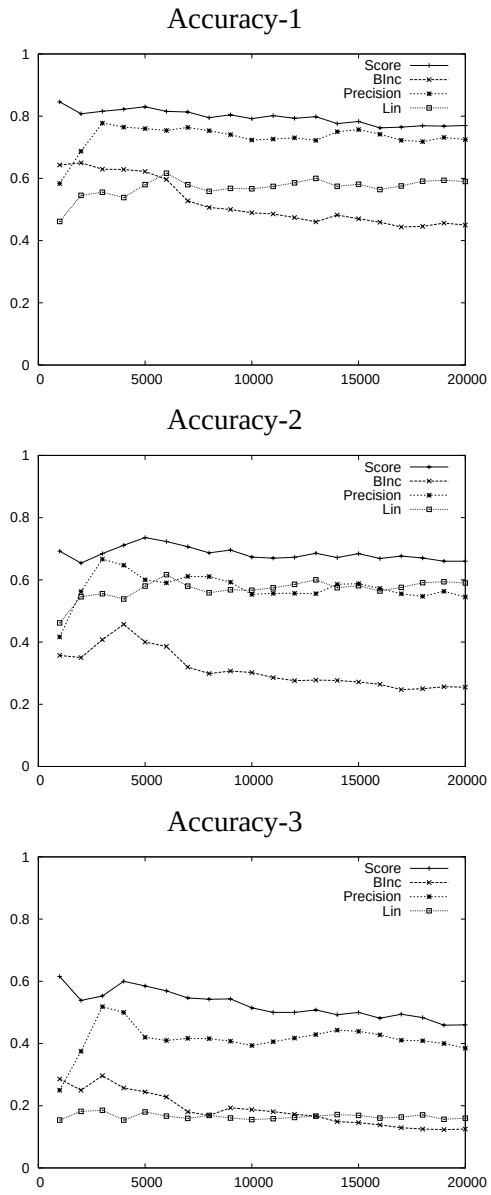


Figure 1: Accuracy of verb entailment pairs.

(5) *suisou-siiku-suru*
tank-raising-do
"raise (fish) in a tank"
→ *siken-houryuu-suru*
test-discharge-do
"stock (with fish) experimentally"

These terms are related in some sense, but they are not entailment pairs.

4.2 Experiment 2: Effectiveness of the Trick

Next, we investigated the effectiveness of the trick described in §3. We evaluated $Score$, $Score_{trick}$, and $Score_{base}$. Table 2 shows the accuracy figures for each method. Figure 2 shows the accuracy figures for the N-best outputs for each method. The

Method	Acc-1	Acc-2	Acc-3
$Score$	**0.770**	**0.660**	**0.460**
$Score_{trick}$	0.725	0.610	0.395
$Score_{base}$	0.590	0.465	0.315

Table 2: Effectiveness of the trick.

results illustrate that introducing the trick significantly improved the performance of $Score_{base}$, and so did multiplying $Score_{trick}$ and $Score_{base}$, which is our proposal $Score$.

(6) shows an example of $Score_{base}$'s errors.

(6) *gazou-sakusei-suru* → *henkou-suru*
image-making-do change-do
"make an image" "change"

This pair has only two shared nouns ($f \in F_l \cap F_r$), and more than 99.99% of the pair's similarity reflects only one of the two. Clearly, the trick would have prevented the pair from being highly ranked.

4.3 Experiment 3: Pairs of Frequent Verbs

We found that the errors made by Lin and BInc in Experiment 1 were mostly pairs of infrequent verbs such as technical terms. Thus, we conducted the acquisition of entailment pairs targeting more frequent verbs to see how their performance changed. The experimental conditions were the same as in Experiment 1, except that the templates ($\langle p, v \rangle$) used were all $Freq(\langle p, v \rangle) \geq 200$.

Table 3 shows the accuracy figures for each method with the changes in accuracy from those of the original methods in parentheses. The results show that the accuracies of $Score$ and Precision (the two best methods in Experiment 1) degraded, while the other two improved a little. We suspect that the performance difference between these methods would get smaller if we further restricted the target verbs to more frequent ones.

Method	Acc-1	Acc-2	Acc-3
$Score$	0.690 (−0.080)	0.520 (−0.140)	0.335 (−0.125)
BInc	0.455 (+0.005)	0.295 (+0.040)	0.160 (+0.035)
Precision	0.450 (−0.275)	0.355 (−0.190)	0.205 (−0.180)
Lin	0.635 (+0.045)	0.385 (+0.015)	0.205 (+0.045)

Table 3: Accuracy of frequent verb pairs.

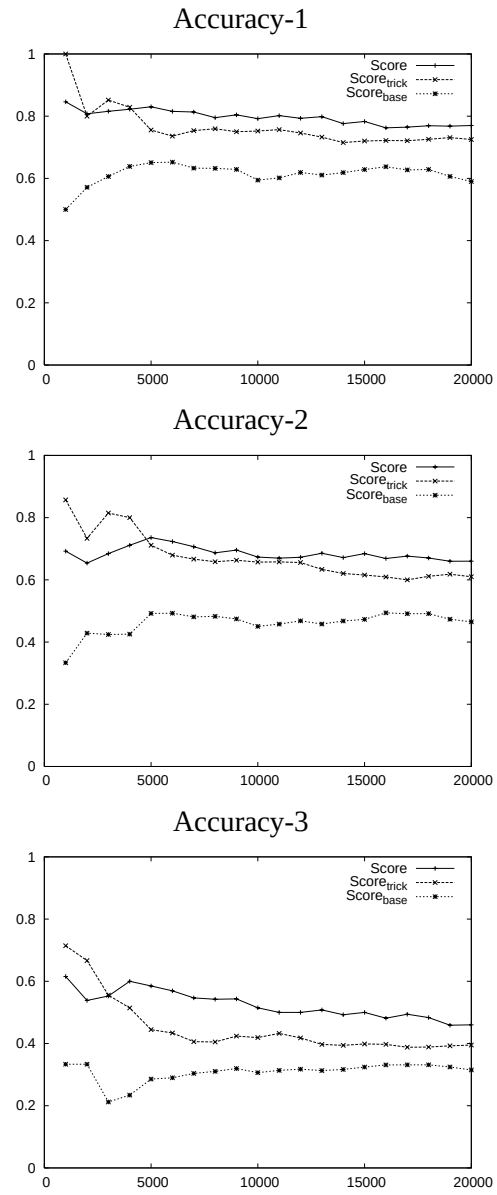


Figure 2: Accuracy of verb entailment pairs acquired by $Score$, $Score_{trick}$, and $Score_{base}$.

However, we believe that dealing with verbs comprehensively, including infrequent ones, is important, since, in the era of information explosion, the impact on applications is determined not only by frequent verbs but also infrequent ones that constitute the long tail of a verb-frequency graph. Thus, this tendency does not matter for our purpose.

4.4 Experiment 4: Template Pairs

This section presents the entailment acquisition accuracy for template pairs to show that our method can also perform the entailment acquisition of unary templates. We presented pairs of unary templates, obtained by the procedure in

§3.4, to the annotators. In doing so, we restricted the correct entailment pairs to those for which entailment always held regardless of what argument filled the two unary slots, and the two slots had to be filled with the same argument, as exemplified in (2). We evaluated $Score$ and Precision.

Table 4 shows the accuracy of the acquired pairs of unary templates. Compared to verb entailment

	Method	Acc-1	Acc-2	Acc-3
Scoring Slots	$Score$	**0.655** (−0.115)	**0.510** (−0.150)	**0.300** (−0.160)
	Precision	0.565 (−0.160)	0.430 (−0.115)	0.265 (−0.120)
Nom Slots	$Score$	**0.665** (−0.105)	**0.515** (−0.145)	**0.315** (−0.145)
	Precision	0.490 (−0.235)	0.325 (−0.220)	0.215 (−0.170)

Table 4: Accuracy of entailment pairs of templates whose slots were used for scoring.

acquisition, the accuracy of both methods dropped by about 10%. This was mainly due to the evaluation restriction exemplified in (2) which was not introduced in the previous experiments; the annotators ignored the argument correspondence between the verb pairs in Experiment 1. Also note that $Score$ outperformed Precision in this experiment, too.

(7) and (8) are examples of the Scoring Slots template entailment pairs and (9) is that of the Nom Slots acquired by our method.

(7) *X-wo tatigui-suru* → *X-wo taberu*
X-ACC standing.up.eating-do X-ACC eat
"eat X standing up" "eat X"

(8) *X-de marineedo-suru* → *X-wo ireru*
X-INS marinade-do X-ACC pour
"marinate with X" "pour X"

(9) *X-ga NBA-iri-suru* · · · (was *X-de* (INS))
X-NOM NBA-entering-do
'X joins an NBA team'
→ *X-ga nyuudan-suru* · · · (was *X-de*)
X-NOM enrollment-do
"X joins a team"

4.5 Experiment 5: Verb Pairs form the Top 100,000

Finally, we examined the accuracy of the top 100,000 verb pairs acquired by $Score$ and Precision. As Table 5 shows, $Score$ outperformed Precision. Note also that $Score$ kept a reasonable accuracy for the top 100,000 results (Acc-2: 48%). The accuracy is encouraging enough to consider human annotation for the top 100,000 results to produce a language resource for verb entailment, which we actually plan to do.

Method	Acc-1	Acc-2	Acc-3
$Score$	**0.610**	**0.480**	**0.300**
Precision	0.470	0.295	0.190

Table 5: Accuracy of the top 100,000 verb pairs.

Below are correct verb entailment examples from the top 100,000 results of our method.

(10) The **121**th pair
kaado-kessai-suru → *siharau*
card-payment-do pay
"pay by card" "pay"

(11) The **6,081**th pair
saitei-suru → *sadameru*
adjudicate-do settle
"adjudicate" "settle"

(12) The **15,464**th pair
eraa-syuuryou-suru → *jikkou-suru*
error-termination-do perform-do
"abend" "execute"

(13) The **30,044**th pair
ribuuto-suru → *kidou-suru*
reboot-do start-do
"reboot" "boot"

(14) The **57,653**th pair
rinin-suru → *syuunin-suru*
resignation-do accession-do
"resign" "accede"

(15) The **70,103**th pair
sijou-tounyuu-suru → *happyou-suru*
market-input-do publication-do
"bring to the market" "publicize"

Below are examples of erroneous pairs from our results. (16) is a causal relation but not an entailment. (17) is a contradictory pair.

(16) The **5,475**th pair
juken-suru → *goukaku-suru*
take.an.exam-do acceptance-do
"take an exam" "gain admission"

(17) The **40,504**th pair

ketujou-suru	→	*syutujou-suru*
not.take.part-do		take.part-do
"not take part"		"take part"

5 Conclusion

This paper addressed verb entailment acquisition from the Web, and proposed a novel directional similarity measure $Score$. Through a series of experiments, we showed **i)** that $Score$ outperforms the previously proposed measures, Lin, Precision, and BInc in large scale verb entailment acquisition, **ii)** that our proposed trick implemented in $Score_{trick}$ significantly improves the accuracy of verb entailment acquisition despite its simplicity, **iii)** that $Score$ worked better than the others even when we restricted the target verbs to more frequent ones, **iv)** that our method is also moderately successful at producing template-level entailment pairs, and **v)** that our method maintained reasonable accuracy (in terms of human annotation) for the top 100,000 results. As examples of the acquired verb entailment pairs illustrated, our method can acquire from an ocean of information, namely the Web, a variety of verb entailment pairs ranging from those that are used in daily life to those that are used in very specific fields.

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL2005)*, pages 597–604.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of HLT-NAACL 2003*, pages 16–23.

Rahul Bhagat, Patrick Pantel, and Eduard Hovy. 2007. Ledir: An unsupervised algorithm for learning directionality of inference rules. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP2007)*, pages 161–170.

Atsushi Fujita and Satoshi Sato. 2008. A probabilistic model for measuring grammaticality and similarity of automatically generated paraphrases of predicate phrases. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING2008)*, pages 225–232.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL2005)*, pages 107–114.

Ali Ibrahim, Boris Katz, and Jimmy Lin. 2003. Extracting structural paraphrases from aligned monolingual corpora. In *Proceedings of the 2nd International Workshop on Paraphrasing (IWP2003)*, pages 57–64.

Daisuke Kawahara and Sadao Kurohashi. 2006a. Case Frame Compilation from the Web using High-Performance Computing. In *Proceedings of The 5th International Conference on Language Resources and Evaluation (LREC-06)*, pages 1344–1347.

Daisuke Kawahara and Sadao Kurohashi. 2006b. A Fully-Lexicalized Probabilistic Model for Japanese Syntactic and Case Structure Analysis. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL2006)*, pages 176–183.

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

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL1998)*, pages 768–774.

Viktor Pekar. 2006. Acquisition of verb entailment from text. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL2006)*, pages 49–56.

Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of the 2nd international Conference on Human Language Technology Research (HLT2002)*, pages 313–318.

Idan Szpector, Hristo Tanev, Ido Dagan, and Bonaventura Coppola. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP2004)*, pages 41–48.

Idan Szpektor and Ido Dagan. 2008. Learning entailment rules for unary template. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING2008)*, pages 849–856.

Kentaro Torisawa. 2005. Automatic acquisition of expressions representing preparation and utilization of an object. In *Proceedings of the Recent Advances in Natural Language Processing (RANLP05)*, pages 556–560.

Kentaro Torisawa. 2006. Acquiring inference rules with temporal constraints by using japanese coodinated sentences and noun-verb co-occurences. In *Proceedings of the Human Language Technology Conference of the Norh American Chapter of the ACL (HLT-NAACL2006)*, pages 57–64.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP2003)*, pages 81–88.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Maria Teresa Pazienza. 2006. Discovering asymmetric entailment relations between verbs using selectional preferences. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics and 21th International Conference on Computational Linguistics (COLING-ACL2006)*, pages 849–856.

A Syntactified Direct Translation Model with Linear-time Decoding

Hany Hassan
Cairo TDC
IBM
Cairo, Egypt
hanyh@eg.ibm.com

Khalil Sima'an
Language and Computation
University of Amsterdam
Amsterdam, The Netherlands
k.simaan@uva.nl

Andy Way
School of Computing
Dublin City University
Dublin, Ireland
away@computing.dcu.ie

Abstract

Recent syntactic extensions of statistical translation models work with a synchronous context-free or tree-substitution grammar extracted from an automatically parsed parallel corpus. The decoders accompanying these extensions typically exceed quadratic time complexity.

This paper extends the Direct Translation Model 2 (DTM2) with syntax while maintaining linear-time decoding. We employ a linear-time parsing algorithm based on an eager, incremental interpretation of Combinatory Categorial Grammar (CCG). As every input word is processed, the local parsing decisions resolve ambiguity eagerly, by selecting a single supertag–operator pair for extending the dependency parse incrementally. Alongside translation features extracted from the derived parse tree, we explore syntactic features extracted from the incremental derivation process. Our empirical experiments show that our model significantly outperforms the state-of-the art DTM2 system.

1 Introduction

Syntactic structure is gradually showing itself to constitute a promising enrichment of state-of-the-art Statistical Machine Translation (SMT) models. However, it would appear that the decoding algorithms are bearing the brunt of this improvement in terms of time and space complexity. Most recent extensions work with a synchronous context-free or tree-substitution grammar extracted from an automatically parsed parallel corpus. While attractive in many ways, the decoders that are needed for these types of grammars usually have time and space complexities that are far beyond linear. Leaving pruning aside, there is a genuine question as to whether syntactic structure necessarily implies more complex decoding algorithms. This paper shows that this need not necessarily be the case.

In this paper we extend the Direct Translation Model (DTM2) (Ittycheriah and Roukos, 2007) with target language syntax while maintaining linear-time decoding. With this extension we make three novel contributions to SMT. Our first contribution is to define a linear-time syntactic parser that works as incrementally as standard SMT decoders (Tillmann and Ney, 2003; Koehn, 2004a). At every word position in the target language string, this parser spans *at most a single parse-state* to augment the translation states in the decoder. The parse state summarizes previous parsing decisions and imposes constraints on the set of valid future extensions such that a well-formed sequence of parse states unambiguously defines a dependency structure. This approach is based on an *incremental interpretation* of the mechanisms of Combinatory Categorial Grammar (CCG) (Steedman, 2000).

Our second contribution lies in extending the DMT2 model with a novel set of syntactically-oriented feature functions. Crucially, these feature functions concern the derived (partial) dependency structure as well as local aspects of *the derivation process*, including such information as the CCG lexical categories (supertag), the CCG operators and the intermediate parse states. This accomplishment is interesting both from a linguistic and technical point of view.

Our third contribution is the extension of the standard phrase-based decoder with the syntactic structure and definition of new *grammar-specific pruning techniques* that control the size of the search space. Interestingly, because it is eager, the incremental parser used in this work is hard pushed to perform at a parsing level close to state-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1182–1191,
Singapore, 6-7 August 2009.

of-the-art cubic-time parsers. Nevertheless, the parsing information it provides allows for significant improvement in translation quality.

We test the new model, called the Dependency-based Direct Translation Model (DDTM), on standard Arabic–English translation tasks used in the community, including LDC and GALE data. We show that our DDTM system provides significant improvements in BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) scores over the already extremely competitive DTM2 system. We also provide results of manual, qualitative analysis of the system output to provide insight into the quantitative results.

This paper is organized as follows. Section 2 reviews the related work. Section 3 discusses the DTM2 baseline model. Section 4 presents the general workings of the incremental CCG parser laying the foundations for integrating it into DTM2. Section 5 details our own DDTM, the dependency-based extension of the DTM2 model. Section 6 reports on extensive experiments and their results. Section 7 provides translation output to shed further detailed insight into the characteristics of the systems. Finally, Section 8 concludes, and discusses future work.

2 Related Work

In (Marcu et al., 2006), it is demonstrated that 'syntactified' target language phrases can improve translation quality for Chinese–English. A stochastic, top-down transduction process is employed that assigns a joint probability to a source sentence and each of its alternative syntactified translations; this is done by specifying a rewriting process of the target parse-tree into a source sentence.

Likewise, the model in (Zollmann and Venugopal, 2006) extends (Chiang, 2005) by augmenting the hierarchical phrases with syntactic categories derived from parsing the target side of a parallel corpus. They use an existing parser to parse the target side of the parallel corpus in order to extract a syntactically motivated, bilingual synchronous grammar as in (Chiang, 2005).

The above-mentioned approaches for incorporating syntax into Phrase-based SMT (Marcu et al., 2006; Zollmann and Venugopal, 2006) share common drawbacks. Firstly, they are based on syntactic phrase-structure parse trees incorporated into a Synchronous CFG or Tree-Substitution Grammar, which makes for a difficult match with non-constituent phrases that are common within Phrase-based SMT. These approaches usually resort to ad hoc solutions to enrich the non-constituent phrases with syntactic structures. Secondly, they deploy chart-based decoders with a high computational cost compared with the phrase-based beam search decoders, e.g., (Tillmann and Ney, 2003; Koehn, 2004a). Thirdly, due to the large parse space, some of the proposed approaches are forced to employ small language models compared to what is usually used in phrase-based systems. To circumvent these computational limitations, various pruning techniques are usually needed, e.g., (Huang and Chiang, 2007).

Other recent approaches, e.g., (Birch et al., 2007; Hassan et al., 2007; Hassan et al., 2008a) incorporate a linear-time supertagger into SMT to take the role of a syntactic language model alongside the standard language model. While these approaches share with our work the use of lexicalized grammars, they never seek to build a full dependency tree or employ syntactic features in order to directly influence the reordering probabilities in the decoder. In the current work, we expand our previous work in (Hassan et al., 2007; Hassan et al., 2008a) to introduce the capabilities of building a full dependency structure and employing syntactic features to influence the decoding process.

Recently, (Shen et al., 2008) introduced an approach for incorporating a dependency-based language model into SMT. They proposed to extract String-to-Dependency trees from the parallel corpus. As the dependency trees are not constituents by nature, they handle non-constituent phrases as well. While this work is in the same general direction as our work, namely aiming at incorporating dependency parsing into SMT, there remain three major differences. Firstly, (Shen et al., 2008) resorted to heuristics to extract the String-to-Dependency trees, whereas our approach employs the well formalized CCG grammatical theory. Secondly, their decoder works bottom-up and uses a chart parser with a limited language model capability (3-grams), while we build on the efficient, linear-time decoder commonly used in phrase-based SMT. Thirdly, (Shen et al., 2008) deploys the dependency language model to augment the lexical language model probability be-

tween two head words but never seek a full dependency graph. In contrast, our approach integrates an incremental parsing capability, that produces the partial dependency structures incrementally while decoding, and thus provides for better guidance for the search of the decoder for more grammatical output. To the best of our knowledge, our approach is the first to incorporate incremental dependency parsing capabilities into SMT while maintaining the linear-time and -space decoder.

3 Baseline: Direct Translation Model 2

The Direct Translation Model (DTM) (Papineni et al., 1997) employs the *a posteriori* conditional distribution $P(T|S)$ of a target sentence T given a source sentence S, instead of the common inversion into $P(S|T)$ based on the source channel approach (Brown et al., 1990). DTM2, introduced in (Ittycheriah and Roukos, 2007), expresses the phrase-based translation task in a unified log-linear probabilistic framework consisting of three components: (i) a prior conditional distribution $P_0(.|S)$, (ii) a number of feature functions $\phi_i()$ that capture the translation and language model effects, and (iii) the weights of the features λ_i that are estimated under MaxEnt (Berger et al., 1996), as in (1):

$$P(T|S) = \frac{P_0(T,J|S)}{Z} \exp \sum_i \lambda_i \phi_i(T,J,S) \quad (1)$$

Here J is the skip reordering factor for the phrase pair captured by $\phi_i()$ and represents the jump from the previous source word, and Z is the per source sentence normalization term. The prior probability P_0 is the prior distribution for the phrase probability which is estimated using the phrase normalized counts commonly used in conventional Phrase–based SMT systems, e.g., (Koehn et al., 2003).

DTM2 differs from other Phrase–based SMT models in that it extracts from a word-aligned parallel corpus only a *non-redundant* set of *minimal phrases* in the sense that no two phrases overlap with each other.

Baseline DTM2 Features: The baseline employs the following five types of features (beside the language model):

- *Lexical Micro Features* examining source and target words of the phrases,
- *Lexical Context Features* encoding the source and target phrase context (i.e. previous and next source and previous target),
- *Source Morphological Features* encoding morphological and segmentation characteristics of source words.
- *Part-of-Speech Features* encoding source and target POS tags as well as the POS tags of the surrounding contexts of phrases.

The DTM2 approach based on MaxEnt provides a flexible framework for incorporating other available feature types as we demonstrate below.

DTM2 Decoder: The decoder for the baseline is a beam search decoder similar to decoders used in standard phrase-based log-linear systems such as (Tillmann and Ney, 2003) and (Koehn, 2004a). The main difference between the DTM2 decoder and the standard Phrase–based SMT decoders is that DTM2 deploys Maximum Entropy probabilistic models to obtain the translation costs and various feature costs by deploying the features described above in a discriminative MaxEnt fashion.

In the rest of this paper we adopt the DTM2 formalization of translation as a discriminative task, and we describe the CCG-based incremental dependency parser that we use for extending the DTM2 decoder, and then list a new set of syntactic dependency feature functions that extend the DTM2 feature set. We also discuss pruning and other details of the approach.

4 The Incremental Dependency Parser

As it processes an input sentence left-to-right word-by-word, the incremental dependency model builds—for each prefix of the input sentence—a partial parse that is a subgraph of the partial parse that it builds for a longer prefix. The dependency graph is constructed incrementally, in that the subgraph constructed at a preceding step is never altered or revised in any later steps. The following schematic view in (2) exhibits the general workings of this parser:

$$S_0 \xrightarrow[w_1,st_1]{o_1} S_1 \xrightarrow[w_2,st_2]{o_2} S_2 \cdots\cdots S_i \xrightarrow[w_i,st_i]{o_i} S_{i+1} \cdots\cdots S_n \quad (2)$$

The syntactic process is represented by a sequence of transitions between adjacent syntactic states S_i.

A transition from state S_{i-1} to S_i scans the current word w_i and stochastically selects a complex lexical descriptor/category st_i and an operator o_i given the local context in the transition sequence. The syntactic state S_i summarizes all the syntactic information about fragments that have already been processed and registers the syntactic arguments which are to be expected next. Only an impoverished deterministic procedure (called a 'State-Realizer') is needed in order to compose a state S_i with the previous states $S_0 \ldots S_{i-1}$ in order to obtain a *fully connected intermediate dependency structure* at every position in the input.

To implement the incremental parsing scheme described above we use the parser described in (Hassan et al., 2008b; Hassan et al., 2009), which is based on Combinatory Categorial Grammar (CCG) (Steedman, 2000). We only briefly describe this parser as its full description is beyond the scope of this paper. The notions of a *supertag* as a lexical category and the process of *supertagging* are both crucial here (Bangalore and Joshi, 1999). Fortunately, CCG specifies the desired kind of lexical categories (supertags) st_i for every word and a small set of combinatory operators o_i that combine the supertag st_i with a previous parse state S_{i-1} into the next parse state S_i. In terms of CCG representations, the parse state is a CCG composite category which specifies either a functor and the arguments it expects to the right of the current word, or is itself an argument for a functor that will follow it to the right. At the first word in the sentence, the parse state consists solely of the supertag of that word.

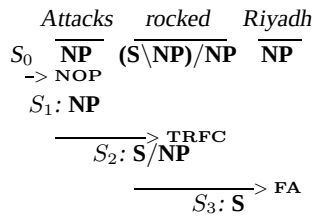


Figure 1: A sentence and possible supertag-, operator- and state-sequences. NOP: No Operation; TRFC: Type Raise-Forward Composition; FA: Forward Application. The CCG operators used show that `Attacks` *and* `Riyadh` *are both dependents of* `rocked`*.*

Figure 1 exhibits an example of the workings of this parser. Practically speaking, after POS tagging the input sentence, the parser employs two components:

- A Supertag-Operator Tagger which proposes a supertag–operator pair for the current word,
- A deterministic State-Realizer, which realizes the current state by applying the current operator to the previous state and the current supertag.

The Supertag-Operator Tagger is a probablistic component while the State-Realizer is a deterministic component. The generative model underlying this component concerns the probability $P(W, S)$ of a word sequence $W = w_1^n$ and a parse-state sequence $S = S_1^n$, with associated supertag sequence $ST = st_1^n$ and operator sequence $O = o_1^n$, which represents a possible derivation. Note that given the choice of supertags st_i and operator o_i, the state S_i is calculated deterministically by the State-Realizer.

A generative version of this model is described in (3):

$$P(W,S) = \prod_{i=1}^{n} \overbrace{P(w_i|W_{i-1}S_{i-1})}^{Word\ Predictor} \cdot \overbrace{P(st_i|W_i)}^{Supertagger} \cdot \overbrace{P(o_i|W_i, S_{i-1}, ST_i)}^{Operator\ Tagger} \quad (3)$$

In (3):

- $P(W, S)$ represents the product of the production probabilities at each parse-state and is similar to the structured language model representation introduced in (Chelba, 2000).
- $P(w_i|W_{i-1}S_{i-1})$ is the probability of w_i given the previous sequence of words W_{i-1} and the previous sequence of states S_{i-1},
- $P(st_i|W_i)$: is the supertag st_i probability given the word sequence W_i up to the current position. Basically, this represents a sequence tagger (a 'supertagger').
- $P(o_i|W_i, S_{i-1}, ST_i)$ represents the probability of the operator o_i given the previous words, supertags and state sequences up to the current position. This represents a CCG operator tagger.

The different local conditional components (for every i) in (3) are estimated as discriminative MaxEnt submodels trained on a corpus of *incremental CCG derivations*. This corpus was extracted from the CCGbank (Hockenmaier, 2003)

by transforming every normal form derivation into strictly left-to-right CCG derivations, with the CCG operators only slightly redesigned to allow incrementality while still satisfying the dependencies in the CCGbank (cf. (Hassan et al., 2008b; Hassan et al., 2009)).

As mentioned before, the State-Realizer is a deterministic function. Starting at the first word with (obviously) a null previous state, the realizer performs the following deterministic steps for each word in turn: (i) set the current supertag and operator to those of the current word; (ii) at the current state, apply the current operator to the previous state and current supertag; (iii) add edges to the dependency graphs between words that are linked as CCG arguments; and (iv) if not at the end of the sentence, set the previous state to the current one, then set the current word to the next one, and iterate from (i).

It is worth noting that the proposed dependency parser is deterministic in the sense that it maintains *only one parse state per word*. This characteristic is crucial for its incorporation into a large-scale SMT system to avoid explosion of the translation space during decoding.

5 Dependency-based DTM (DDTM)

In this section we extend the DTM2 model with incremental target dependency-based syntax. We call the resulting model the Dependency-based Direct Translation Model (DDTM). This extension takes place by (i) extracting syntactically enriched minimal phrase pairs, (ii) including a new set of syntactic feature functions among the exponential model features, and (iii) adapting the decoder for dealing with syntax, including various pruning strategies and enhancements. Next we describe each extension in turn.

5.1 Phrase Table: Incremental Syntax

The target-side sentences in the word-aligned parallel corpus used for training are parsed using the incremental dependency parser described in section 4. This results in a word-aligned parallel corpus where the words of the target sentences are tagged with supertags and operators. From this corpus we extract the set of minimal phrase pairs using the method described in (Ittycheriah and Roukos, 2007), extracting along with every target phrase the associated sequences of supertags and operators. As shown in (4), a source phrase $s_1, \ldots, s_n$ translates into a target phrase $w_1, \ldots, w_m$ where every word w_i is labeled with a supertag st_i, and a possible parsing operator o_i appearing with it in the parsed parallel corpus:

$$s_1 ... s_n \longrightarrow [w_1, st_1, o_1] ... [w_m, st_m, o_m] \quad (4)$$

Hence, our phrase table associates with every target phrase an *incremental parsing subgraph*. These subgraphs along with their probabilities represent our phrase table augmented with incremental dependency parsing structure.

This representation turns the complicated problem of MT with incremental parsing into a sequential classification problem in which the classifier deploys various features from the source sentence and the candidate target translations to specify a sequence of decisions that finally results in an output target string along with its associated dependency graph. The classification decisions are performed in sequence step-by-step while traversing the input string to provide decisions on possible words, supertags, operators and states. A beam search decoder simultaneously decides which sequence is the most probable.

5.2 DDTM Features

The exponential model and the MaxEnt framework used in DTM2 and DDTM enabled us to explore the utility of incremental syntactic parsing within a rich feature space. In our DDTM system, we implemented a set of features alongside the baseline DTM2 features that were discussed in Section 3. The features described here encode all the probabilistic components in (3) within a log linear interpretation along with some more empirically intuitive features.

- Supertag-Word features: these features examine the target phrase words with their associated supertags and is related to the Supertagger component in (3).
- Supertag sequence features: these features encode *n*-gram supertags (equivalent to the *n*-gram supertags Language Model). This feature is related to the supertagger component as well.
- Supertag-Operator features: these features encode supertags and associated operators which is related to the Operator Tagger component in (3).

- Supertag-State features: these features register state and supertag co-occurrences.
- State sequence features: these features encode *n*-gram state features and are equivalent to an *n*-gram Language Model over parse state sequences which is related to the multiplication in (3).
- Word-State sequence features: these features encode words and states co-occurrences which is related to the Word Predictor component in (3).

The exponential model and the MaxEnt framework used in DTM2 and DDTM enable us to explore the utility of incremental syntactic parsing with the use of minimal phrases within a rich feature space.

5.3 DDTM Decoder

In order to support incremental dependency parsing, we extend the DTM2 decoder in three ways: firstly, by constructing the syntactic states during decoding; secondly, by extending the hypothesis structures to incorporate the syntactic states and the partial dependency derivations; and thirdly, by modifying the pruning strategy to handle the large search space.

At decoding time, each hypothesis state is associated with a parse-state which is constructed while decoding using the incremental parsing approach introduced in ((Hassan et al., 2008b; Hassan et al., 2009)). The previous state, the sequences of supertags and CCG incremental operators are deployed in a deterministic manner to realize the parse-states as well as the intermediate dependency graphs between words.

Figure 2 shows the DDTM decoder while decoding a sentence with the English translation "Attacks rocked Riyadh". Each hypothesis is associated with a parse-state S_i and a partial dependency graph (shown for some states only). Moreover, each transition is associated with an operator o that combines the previous state and the current supertag st to construct the next state S_i. The decoder starts from a null state S_1 and then proceeds with a possible expansion with the word "attacks", supertag NP and operator NOP to produce the next hypothesis with state S_2 and category NP. Further expansion for that path with the verb "rocked", supertag '$(S\backslash NP)/NP$ and operator $TRFC$ will produce the state S_5 with category S/NP. The partial dependency graph for state S_5 is shown above the state where a dependency relation between the two words is established. Furthermore, another expansion with the word "Riyadh", supertag NP and operator FA produces state S_7 with category S and a completed dependency graph as shown above the state. Another path which spans the states S_1, S_3 , S_6 and S_8 ends with a state category S/NP and a partial dependency graph as shown under state S_8 where the dependency graph is still missing its object (e.g. "Riyadh attacks rocked the Saudi Govt.").

The addition of parse-states may result in a very large search space due to the fact that the same phrase/word may have many possible supertags and many possible operators. Moreover, the same word sequences may have many parse-state sequences and, therefore, many hypotheses that represent the same word sequence. The search space is definitely larger than the baseline search space. We adopt the following three pruning heuristics to limit the search space.

5.3.1 Grammatical Pruning

Any hypothesis which does not constitute a valid parse-state is discarded, i.e. if the previous parse-state and the current supertag sequence cannot construct a valid state using the associated operator sequence, then the expansion is discarded. Therefore, this pruning strategy maintains only fully connected graphs and discards any partially connected graphs that might result during the decoding process.

As shown in Figure 2, the expansion from state S_1 to state S_4 (with the dotted line) is pruned and not expanded further because the proposed expansion is the verb "attacks", supertag $(S\backslash NP)/NP$ and operator $TRFC$. Since the previous state is NULL, it cannot be combined with the verb using the $TRFC$ operator. This would produce an undefined state and thus the hypothesis is discarded.

5.3.2 Supertags and Operators Threshold

We limit the supertag and operator variants per target phrase to a predefined number of alternatives. We tuned this on the MT03 DevSet for the best accuracy while maintaining a manageable search space. The supertags limit was set to four alternatives while the operators limit was set to three.

As shown in Figure 2, each word can have many alternatives with different supertags. In this example the word "attacks" has two forms, namely a

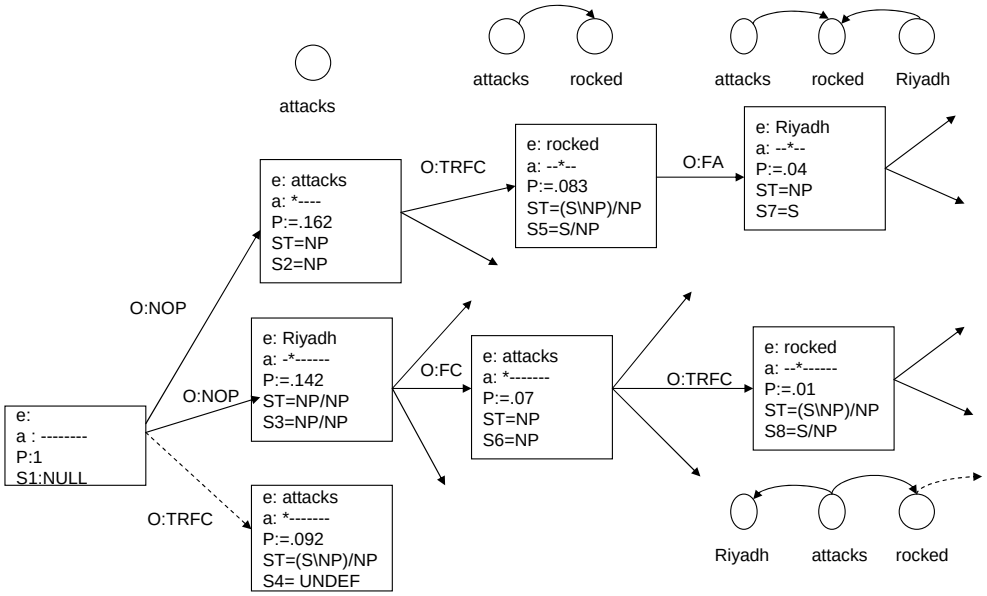


Figure 2: DDTM Decoder: each hypothesis has a parse state and a partial dependency structure.

noun and a verb, with different supertags and operators. The proposed thresholds limit the possible alternatives to a reasonable number.

5.3.3 Merging Hypotheses

Standard Phrase–based SMT decoders merge translation hypotheses if they cover the same source words and share the same *n*-gram language model history. Similarly, DDTM decoder merges translation hypotheses if they cover the same source words, share the same *n*-gram language model history and share the same parse-state history. This helps in reducing the search space by merging paths that will not constitute a part of the best path.

6 Experiments

We conducted experiments on an Arabic-to-English translation task using LDC parallel data and GALE parallel data. We used the UN parallel corpus and LDC news corpus together with the GALE parallel corpus, totaling 7.8M parallel sentences. The 5-gram Language Model was trained on the English Gigaword Corpus and the English part of the parallel corpus. Our baseline system is similar to the system described in (Ittycheriah and Roukos, 2007). We report results on NIST MT05 and NIST MT06 evaluations test sets using BLEU and TER as automatic evaluation metrics.

To train the DDTM model, we use the incremental parser introduced in (Hassan et al., 2008b; Hassan et al., 2009) to parse the target side of the parallel training data. Each sentence is associated with supertag, operator and parse-state sequences. We then train models with different feature sets.

Results: We compared the baseline DTM2 (Ittycheriah and Roukos, 2007) with our DDTM system with the features listed above. We examine the effect of all features on system performance. In this set of experiments we used LDC parallel data only which is composed of 3.7M sentences and the results are reported on MT05 test set. Each of the examined systems deploys DTM2 features in addition to a number of newly added syntactic features. The systems examined are:

- DTM2: Direct Translation model 2 baseline.
- D-SW: DTM2 + Supertag-Word features.
- D-SLM: DTM2 + Supertag-Word and supertag *n*-gram features.
- D-SO: DTM2+ Supertag-Operator features.
- D-SS : DTM2 + supertags and states features with parse-state construction.
- D-WS : DTM2 + words and states features with parse-state construction.
- D-STLM: DTM2 + state *n*-gram features with parse-state construction.
- DDTM: fully fledged system with all features that proved useful above which are: Supertag-Word features, supertag *n*-gram

features, supertags and states features and state *n*-gram features .

System	BLEU Score on MT05
DTM2-Baseline	52.24
D-SW	52.28
D-SLM	52.29
D-SO	52.01
D-SS	52.39
D-WS	52.03
D-STLM	52.53
DDTM	**52.61**

Table 1: DDTM Results with various features.

As shown in Table 1, the DTM baseline system demonstrates a very high BLEU score, unsurprisingly given its top-ranked performance in two recent major MT evaluation campaigns. Among the features we tried, supertags and *n*-gram supertags systems (D-SW and D-SLM systems) give slight yet statistically insignificant improvements. On the other hand, the states *n*-gram sequence features (D-SS and DDTM systems) give small yet statistically significant improvements (as calculated via bootstrap resampling (Koehn, 2004b)). The D-WS system shows a small degradation in performance, probably due to the fact that the states-words interactions are quite sparse and could not be estimated with good evidence. Similarly, the D-SO system shows a small degradation in performance. When we investigated the features types, we found out that all features that deploy the operators had bad effect on the model. We think this is due to the fact that the operator set is a small set with high evidence in many training instances such that it has low discriminative power on it is own. However, it implicitly helps in producing the state sequence which proved useful.

System	DTM2-Baseline	DDTM
MT05 (BLEU)	55.28	55.66
MT05 (*TER*)	38.79	38.48
MT06 (BLEU)	43.56	43.91
MT06 (*TER*)	49.08	48.65

Table 2: DDTM Results on MT05 and MT06.

We examined a combination of the best features in our DDTM system on a larger training data comprising 7.8M sentences from both NIST and GALE parallel corpora. Table 2 shows the results on both MT05 and MT06 test sets. As shown, DDTM significantly outperforms the state-of-the-art baseline system. It is worth noting that DDTM outperforms this baseline even when very large amounts of training data are used. Despite the fact that the actual scores are not so different, we found that the baseline translation output and the DDTM translation outout are significantly different. We measured this by calculating the TER between the baseline translation and the DDTM translation for the MT05 test set, and found this to be 25.9%. This large difference has not been realized by the BLEU or TER scores in comparison to the baseline. We believe that this is due to the fact that most changes that match the syntactic constraints do not bring about the best match where the automatic evaluation metrics are concerned. Accordingly, in the next section we describe the outcome of a detailed manual analysis of the output translations.

7 Manual Analysis of Results

Although the BLEU score does not mark a large improvement by the dependency-based system over the baseline system, human inspection of the data gives us important insights into the pros and cons of the dependency-based model. We analyzed a randomly selected set of 100 sentences from the MT05 test set. In this sample, the baseline and the DDTM system perform similarly in 68% of the sentences. The outputs of both system are similar though not identical. In these cases, the systems may choose equivalent paraphrases. However, the translations using syntactic structures are rather similar. It is worth noting that the DDTM system tends to produce more concise syntactic structures which may lead to less BLUE score due to penalizing the translation length although the translation might be equivelent to the baseline if not better.

In 28% of the sentences, the DDTM system produces remarkably better translations. The examples here illustrate the behaviour of the baseline and the DDTM systems which can be observed consistently throughout the test set. We only highlight some of the examples for illustration purposes. DDTM manages to insert verbs which are deleted by any standard phrase-based SMT system. DDTM prefers to deploy verbs since they have complex and more detailed syntactic structures which give better and more likely state se-

quences. Furthermore, the DDTM system avoids longer noun phrases and instead uses some prepositions in-between. Again, this is probably due to the fact that like verbs, prepositions have a complex syntactic description that give rise to more likely state sequences.

On the other hand, the baseline produced better translation in 8% of the analysis sample. We observed that the baseline is doing better mainly in two cases. The first when the produced translation is very poor and producing poor sysntatctic structure due to out of vocabularies or hard to translate sentences. The second case is with sentences with long noun phrases, in such cases the DDTM system prefres to introduce verbs or prepositions in the middle of long noun phrase and thus the baseline would produce better translations. This is maybe due to the fact that noun phrases have relatively simple structure in CCG such that it did not help in constructing long noun phrases.

Source: وخضع بعد ذلك لفحوصات اجراهَا احد اطباء الشرطة
Reference: *He then underwent medical examinations by a police doctor .*
Baseline: *He was subjected after that tests conducted by doctors of the police .*
DDTM: *Then he underwent tests conducted by doctors of the police .*

Source: وقد هز الرياض مساء اليوم هجومان بسيارتين مفخـختين
Reference: *Riyadh was rocked tonight by two car bomb attacks..*
Baseline: *Riyadh rocked today night attacks by two booby - trapped cars.*
DDTM: *Attacks rocked Riyadh today evening in two car bombs.*

Figure 3: DDTM provides better syntactic structure with more concise translations.

Figure 3 shows two examples where DDTM provides better and more concise syntactic structure. As we can see, there is not much agreement between the reference and the proposed translation. However, longer translations enhance the possibility of picking more common *n*-gram matches via the BLEU score and so increases the chance of better scores. This well-known bias does not favour the more concise output derived by our DDTM system, of course.

8 Conclusion and Future Work

In this paper, we presented a novel model of dependency phrase-based SMT which integrates incremental dependency parsing into the translation model while retaining the linear decoding assumed in conventional Phrase–based SMT systems. To the best of our knowledge, this model constitutes the first effective attempt at integrating a linear-time dependency parser that builds a connected tree incrementally into SMT systems with linear-time decoding. Crucially, it turns out that incremental dependency parsing based on lexicalized grammars such as CCG and LTAG can provide valuable incremental parsing information to the decoder even if their output is imperfect. We believe this robustness in the face of imperfect parser output to be a property of the probabilistic formulation and statistical estimation used in the Direct Translation Model. A noteworthy aspect of our proposed approach is that it integrates features from the derivation process as well as the derived tree. We think that this is possible due to the importance of the notion of a derivation in linguistic frameworks such as CCG and LTAG.

Future work will attempt further extensions of our DDTM system to allow for the exploitation of long-range aspects of the dependency structure. We will work on expanding the features set of DDTM system to leverage features from the constructed dependency structure itself. Finally, we will work on enabling the deployment of source side dependency structures to influence the construction of the target dependency structure based on a bilingually enabled dependency parsing mechanism using the discriminative modeling capabilities.

Acknowledgments

We would like to thank Salim Roukos, IBM TJ Watson Research Center, for fruitful, insightful discussions and for his support during this work. We also would like to thank Abe Ittycheriah, IBM TJ Watson Research Center, for providing the DTM2 baseline and for his support during developing this system. Finally, we would like to thank the anonymous reviewers for their helpful and constructive comments.

References

Bangalore, S. and Joshi, A. (1999). "Supertagging: An Approach to Almost Parsing", *Computational Linguistics* **25**(2):237–265, 1999.

Berger, A. and Della Pietra, S. and Della Pietra, V.J. (1996). Maximum Entropy Approach to Natural Language Processing *Computational Linguistics*, **22**(1): 39–71, 1996.

Birch, A., Osborne, M. and Koehn, P. (2007). CCG Supertags in Factored Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation, ACL-07*, pp.9–16, 2007.

Brown, P., Cocke,J., Della Pietra, S., Jelinek, F., Della Pietra, V.J. Lafferty, R. Mercer and Roossin, P. "A Statistical Approach to Machine Translation" *Computational Linguistics* **16**(2):79–85, 1990.

Chelba, C. (2000). Exploiting Syntactic Structure for Natural Language Modeling. PhD thesis, Johns Hopkins University, Baltimore, MD.

Chiang, D. (2005). A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL05)*, pp.263–270, Ann Arbor, MI.

Hassan, H., Sima'an, K., and Way, A.. (2009). Lexicalized Semi-Incremental Dependency Parsing. In *Proceedings of RANLP 2009, the International Conference on Recent Advances in Natural Language Processing*, Borovets, Bulgaria (to appear).

Hassan, H., Sima'an, K., and Way, A. (2008a). Syntactically Lexicalized Phrase-Based Statistical Translation. *IEEE Transactions on Audio, Speech and Language Processing*, **6**(7):1260–1273.

Hassan, H., Sima'an, K., and Way, A.. (2008b). A Syntactic Language Model Based on Incremental CCG Parsing. In *Proceedings IEEE Workshop on Spoken Language Technology (SLT) 2008*, Goa, India.

Hassan, H., Sima'an, K., and Way, A. (2007). Integrating Supertags into Phrase-based Statistical Machine Translation. In *Proceedings of the ACL-2007, Prague, Czech Republic, pp.288–295, 2007*.

Hockenmaier, J. (2003). *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*, Ph.D Thesis, University of Edinburgh, UK, 2003.

Huang, L. and Chiang, D. (2007). Forest Rescoring: Faster Decoding with Integrated Language Models. In *Proceedings of the ACL-2007, Prague, Czech Republic, 2007*.

Ittycheriah, A. and Roukos, S. (2007). Direct translation model 2. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pp.57–64, Rochester, NY.

Koehn, P. (2004a). Pharaoh: A Beam Search Decoder for phrase-based Statistical Machine Translation Models. Machine Translation: From Real Users to Research. In *Proceedings of 6th Conference of the Association for Machine Translation in the Americas, AMTA 2004*, pp.115–124, Washington, DC.

Koehn, P. (2004b). Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp.388–395, Edmonton, AB, Canada.

Koehn, P. Och, F.J. and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the Joint Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, pp.127–133, Edmonton, AL, Canada.

Marcu, D., Wang, W., Echihabi, A., and Knight, K. (2006). SPMT: Statistical Machine Translation with Syntactified Target Language Phrases. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pp.44–52, Sydney, Australia.

Papineni, K., Roukos, S., and Ward, T. (1997). Feature-Based Language Understanding. In *Proceedings of 5th European Conference on Speech Communication and Technology EUROSPEECH '97* , pp.1435–1438, Rhodes, Greece.

Papineni, K., Roukos, S., Ward, T. and Zhu, W-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pp.311–318, Philadelphia, PA.

Shen, L., Xu, J., and Weischedel, R. (2008). A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pp.577–585, Columbus, OH.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L. and Makhoul, J. (2006) A Study of Translation Edit Rate with Targeted Human Annotation. In *AMTA 2006: Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pp.223–231, Cambridge, MA.

Steedman, M. (2000). *The Syntactic Process*. MIT Press, Cambridge, MA.

Tillmann, C. and Ney, H. (2003). Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, 29(1):97–133.

Zollmann, A. and Venugopal, A. (2006). Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation, HLT/NAACL*, pp.138–141, New York, NY.

Cross-lingual Semantic Relatedness Using Encyclopedic Knowledge

Samer Hassan and **Rada Mihalcea**
Department of Computer Science
University of North Texas
samer@unt.edu, rada@cs.unt.edu

Abstract

In this paper, we address the task of cross-lingual semantic relatedness. We introduce a method that relies on the information extracted from Wikipedia, by exploiting the interlanguage links available between Wikipedia versions in multiple languages. Through experiments performed on several language pairs, we show that the method performs well, with a performance comparable to monolingual measures of relatedness.

1 Motivation

Given the accelerated growth of the number of multilingual documents on the Web and elsewhere, the need for effective multilingual and cross-lingual text processing techniques is becoming increasingly important. In this paper, we address the task of *cross-lingual semantic relatedness*, and introduce a method that relies on Wikipedia in order to calculate the relatedness of words across languages. For instance, given the word *factory* in English and the word *lavoratore* in Italian (En. *worker*), the method can measure the relatedness of these two words despite the fact that they belong to two different languages.

Measures of cross-language relatedness are useful for a large number of applications, including cross-language information retrieval (Nie et al., 1999; Monz and Dorr, 2005), cross-language text classification (Gliozzo and Strapparava, 2006), lexical choice in machine translation (Och and Ney, 2000; Bangalore et al., 2007), induction of translation lexicons (Schafer and Yarowsky, 2002), cross-language annotation and resource projections to a second language (Riloff et al., 2002; Hwa et al., 2002; Mohammad et al., 2007).

The method we propose is based on a measure of closeness between concept vectors automatically built from Wikipedia, which are mapped via the Wikipedia interlanguage links. Unlike previous methods for cross-language mapping, which are typically limited by the availability of bilingual dictionaries or parallel texts, the method proposed in this paper can be used to measure the relatedness of word pairs in any of the 250 languages for which a Wikipedia version exists.

The paper is organized as follows. We first provide a brief overview of Wikipedia, followed by a description of the method to build concept vectors based on this encyclopedic resource. We then show how these concept vectors can be mapped across languages for a cross-lingual measure of word relatedness. Through evaluations run on six language pairs, connecting English, Spanish, Arabic and Romanian, we show that the method is effective at capturing the cross-lingual relatedness of words, with results comparable to the monolingual measures of relatedness.

2 Wikipedia

Wikipedia is a free online encyclopedia, representing the outcome of a continuous collaborative effort of a large number of volunteer contributors. Virtually any Internet user can create or edit a Wikipedia webpage, and this "freedom of contribution" has a positive impact on both the quantity (fast-growing number of articles) and the quality (potential errors are quickly corrected within the collaborative environment) of this online resource.

The basic entry in Wikipedia is an *article* (or *page*), which defines and describes an entity or an event, and consists of a hypertext document with hyperlinks to other pages within or outside Wikipedia. The role of the hyperlinks is to guide the reader to pages that provide additional information about the entities or events mentioned in an article. Articles are organized into *categories*, which in turn are organized into hierarchies. For instance, the article *automobile* is included in the category *vehicle*, which in turn has a parent cate-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1192–1201,
Singapore, 6-7 August 2009.

Language	Articles	Users
English	2,221,980	8,944,947
German	864,049	700,980
French	765,350	546,009
Polish	579,170	251,608
Japanese	562,295	284,031
Italian	540,725	354,347
Dutch	519,334	216,938
Portuguese	458,967	503,854
Spanish	444,696	966,134
Russian	359,677	226,602

Table 1: Top ten largest Wikipedias

gory named *machine*, and so forth.

Each article in Wikipedia is uniquely referenced by an identifier, consisting of one or more words separated by spaces or underscores and occasionally a parenthetical explanation. For example, the article for *bar* with the meaning of *"counter for drinks"* has the unique identifier *bar (counter)*.

Wikipedia editions are available for more than 250 languages, with a number of entries varying from a few pages to two millions articles or more per language. Table 1 shows the ten largest Wikipedias (as of December 2008), along with the number of articles and approximate number of contributors.[1]

Relevant for the work described in this paper are the *interlanguage links*, which explicitly connect articles in different languages. For instance, the English article for *bar (unit)* is connected, among others, to the Italian article *bar (unitá di misura)* and the Polish article *bar (jednostka)*. On average, about half of the articles in a Wikipedia version include interlanguage links to articles in other languages. The number of interlanguage links per article varies from an average of five in the English Wikipedia, to ten in the Spanish Wikipedia, and as many as 23 in the Arabic Wikipedia.

3 Concept Vector Representations using Explicit Semantic Analysis

To calculate the cross-lingual relatedness of two words, we measure the closeness of their concept vector representations, which are built from Wikipedia using explicit semantic analysis (ESA).

Encyclopedic knowledge is typically organized into concepts (or topics), each concept being further described using definitions, examples, and possibly links to other concepts. ESA (Gabrilovich and Markovitch, 2007) relies on the distribution of words inside the encyclopedic descriptions, and builds semantic representations for a given word in the form of a vector of the encyclopedic concepts in which the word appears. In this vector representation, each encyclopedic concept is assigned with a weight, calculated as the term frequency of the given word inside the concept's article.

Formally, let C be the set of all the Wikipedia concepts, and let a be any content word. We define $\vec{a}$ as the ESA concept vector of term a:

$$\vec{a} = \{w_{c_1}, w_{c_2} ... w_{c_n}\}, \quad (1)$$

where w_{c_i} is the weight of the concept c_i with respect to a. ESA assumes the weight w_{c_i} to be the term frequency tf_i of the word a in the article corresponding to concept c_i.

We use a revised version of the ESA algorithm. The original ESA semantic relatedness between the words in a given word pair $a - b$ is defined as the cosine similarity between their corresponding vectors:

$$Relatedness(a, b) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \left\|\vec{b}\right\|}. \quad (2)$$

To illustrate, consider for example the construction of the ESA concept vector for the word *bird*. The top ten concepts containing this word, along with the associated weight (calculated using equation 7), are listed in table 2. Note that the the ESA vector considers all the possible senses of *bird*, including *Bird* as a surname as in e.g., *"Larry Bird."*

Weight	Wikipedia concept
51.4	Lists Of Birds By Region
44.8	Bird
40.3	British Birds Rarities Committee
32.8	Origin Of Birds
31.5	Ornithology
30.1	List Of Years In Birding And Ornithology
29.8	Bird Vocalization
27.4	Global Spread Of H5n1 In 2006
26.5	Larry Bird
22.3	Birdwatching

Table 2: Top ten Wikipedia concepts for the word "bird"

In our ESA implementation, we make three changes with respect to the original ESA algorithm. First, we replace the cosine similarity with

[1] http://meta.wikimedia.org/wiki/List_of_Wikipedias#Grand_Total

a Lesk-like metric (Lesk, 1986), which places less emphasis on the distributional differences between the vector weights and more emphasis on the overlap (mutual coverage) between the vector features, and thus it is likely to be more appropriate for the sparse ESA vectors, and for the possible asymmetry between languages. Let a and b be two terms with the corresponding ESA concept vectors $\vec{A}$ and $\vec{B}$ respectively. Let A and B represent the sets of concepts with a non-zero weight encountered in $\vec{A}$ and $\vec{B}$ respectively. The coverage of $\vec{A}$ by $\vec{B}$ is defined as:

$$G(\vec{B}|\vec{A}) = \sum_{i \in B} w_{a_i} \quad (3)$$

and similarly, the coverage of $\vec{B}$ by $\vec{A}$ is:

$$G(\vec{A}|\vec{B}) = \sum_{i \in A} w_{b_i} \quad (4)$$

where wa_i and wb_i represent the weight associated with concept c_i in vectors $\vec{A}$ and $\vec{B}$ respectively. By averaging these two asymmetric scores, we redefine the relatedness as:

$$Relatedness(a,b) = \frac{G(\vec{B}|\vec{A}) + G(\vec{A}|\vec{B})}{2} \quad (5)$$

Second, we refine the ESA weighting schema to account for the length of the articles describing the concept. Since some concepts have lengthy descriptions, they may be favored due to their high term frequencies when compared to more compact descriptions. To eliminate this bias, we calculate the weight associated with a concept c_i as follows:

$$w_{c_i} = tf_i \times log(M/\,|c_i|), \quad (6)$$

where tf_i represents the term frequency of the word a in concept c_i, M is a constant representing the maximum vocabulary size of Wikipedia concepts, and $|c_i|$ is the size of the vocabulary used in the description of concept c_i.

Finally, we use the Wikipedia category graph to promote category-type concepts in our feature vectors. This is done by scaling the concept's weight by the inverse of the distance d_i to the root category. The concepts that are not categories are treated as leaves, and therefore their weight is scaled down by the inverse of the maximum depth in the category graph. The resulting weighting scheme is:

$$w_{c_i} = tf_i \times log(M/\,|c_i|)/d_i \quad (7)$$

4 Cross-lingual Relatedness

We measure the relatedness of concepts in different languages by using their ESA concept vector representations in their own languages, along with the Wikipedia interlanguage links that connect articles written in a given language to their corresponding Wikipedia articles in other languages. For example, the English Wikipedia article *moon* contains interlanguage links to قمر in the Arabic Wikipedia, *luna* in the Spanish Wikipedia, and *lună* in the Romanian Wikipedia. The interlanguage links can map concepts across languages, and correspondingly map concept vector representations in different languages.

Formally, let C_x and C_y be the sets of all Wikipedia concepts in languages x and y, with corresponding translations in the y and x languages, respectively. If $tr_{xy}()$ is a translation function that maps a concept $c_i \in C_x$ into the concept $c_i^{'} \in C_y$ via the interlanguage links, we can write:

$$tr_{xy}(c_i) = c_i^{'}, \quad (8)$$

The projection of the ESA vector $\vec{t}$ from language x onto y can be written as:

$$tr_{xy}(\vec{t}) = \left\{ w_{tr_{xy}(c_1)} ... w_{tr_{xy}(c_n)} \right\}. \quad (9)$$

Using equations 5, 7, and 9, we can calculate the cross-lingual semantic relatedness between any two content terms a_x and b_y in given languages x and y as:

$$sim(a_x, b_y) = \frac{G(tr_{yx}(\vec{B})|\vec{A}) + G(\vec{A}|tr_{yx}(\vec{B}))}{2}. \quad (10)$$

Note that the weights assigned to Wikipedia concepts inside the concept vectors are language specific. That is, two Wikipedia concepts from different languages, mapped via an interlanguage link, can, and often do have different weights.

Intuitively, the relation described by the interlanguage links should be reflective and transitive. However, due to Wikipedia's editorial policy, which accredits users with the responsibility

of maintaining the articles, these properties are not always met. Table 3 shows real cases where the transitive and the reflective properties fail due to missing interlanguage links.

Relation	Exists
Reflectivity	
Kafr-El-Dawwar Battle(en) ↦ معركة كفر الدوّار(ar)	Yes
معركة كفر الدوّار(ar) ↦ Kafr-El-Dawwar Battle(en)	No
Transitivity	
Intifada(en) ↦ Intifada(es)	Yes
Intifada(es) ↦ انتفَاضة(ar)	Yes
Intifada(en) ↦ انتفَاضة(ar)	No

Table 3: Reflectivity and transitivity in Wikipedia

We solve this problem by iterating over the translation tables and extracting all the missing links by enforcing the reflectivity and the transitivity properties. Table 4 shows the initial number of interlanguage links and the discovered links for the four languages used in our experiments. The table also shows the coverage of the interlanguage links, measured as the ratio between the total number of interlanguage links (initial plus discovered) originating in the source language towards the target language, divided by the total number of articles in the source language.

	Interlanguage links		
Language pair	Initial	Discov.	Cover.
English → Spanish	293,957	12,659	0.14
English → Romanian	86,719	4,641	0.04
English → Arabic	56,233	3,916	0.03
Spanish → English	294,266	7,328	0.58
Spanish → Romanian	39,830	3,281	0.08
Spanish → Arabic	33,889	3,319	0.07
Romanian → English	75,685	6,783	0.46
Romanian → Spanish	36,002	3,546	0.22
Romanian → Arabic	15,777	1,698	0.10
Arabic → English	46,072	3,170	0.33
Arabic → Spanish	28,142	3,109	0.21
Arabic → Romanian	15,965	1,970	0.12

Table 4: Interlanguage links (initial and discovered) and their coverage in Wikipedia versions in four languages.

5 Experiments and Evaluations

We run our experiments on four languages: English, Spanish, Romanian and Arabic. For each of these languages, we use a Wikipedia download from October 2008. The articles were preprocessed using Wikipedia Miner (Milne, 2007) to extract structural information such as generality, and interlanguage links. Furthermore, articles were also processed to remove numerical content, as well as any characters not included in the given language's alphabet. The content words are stemmed, and words shorter than three characters are removed (a heuristic which we use as an approximation for stopword removal). Table 5 shows the number of articles in each Wikipedia version and the size of their vocabularies, as obtained after the pre-processing step.

	Articles	Vocabulary
English	2,221,980	1,231,609
Spanish	520,154	406,134
Arabic	149,340	216,317
Romanian	179,440	623,358

Table 5: Number of articles and size of vocabulary for the four Wikipedia versions

After pre-processing, the articles are indexed to generate the ESA concept vectors. From each Wikipedia version, we also extract other features including article titles, interlanguage links, and Wikipedia category graphs. The interlanguage links are further processed to recover any missing links, as described in the previous section.

5.1 Data

For the evaluation, we build several cross-lingual datasets based on the standard Miller-Charles (Miller and Charles, 1998) and WordSimilarity-353 (Finkelstein et al., 2001) English word relatedness datasets.

The Miller-Charles dataset (Miller and Charles, 1998) consists of 30-word pairs ranging from synonymy pairs (e.g., *car - automobile*) to completely unrelated terms (e.g., *noon - string*). The relatedness of each word pair was rated by 38 human subjects, using a scale from 0 (not-related) to 4 (perfect synonymy). The dataset is available only in English and has been widely used in previous semantic relatedness evaluations (e.g., (Resnik, 1995; Hughes and Ramage, 2007; Zesch et al., 2008)).

The WordSimilarity-353 dataset (also known as Finkelstein-353) (Finkelstein et al., 2001) consists of 353 word pairs annotated by 13 human experts, on a scale from 0 (unrelated) to 10 (very closely related or identical). The Miller-Charles set is a subset in the WordSimilarity-353 data set. Unlike the Miller-Charles data set, which consists only of

	Word pair		
English	coast - shore	car - automobile	brother - monk
Spanish	costa - orilla	coche - automovil	hermano - monje
Arabic	شَاطيء - سَاحل	عربه - سيَارة	رَاهب - شقيق
Romanian	ţărm - mal	maşfină - automobil	frate - călugăr

Table 6: Word pair translation examples

single words, the WordSimilarity-353 set also features phrases (e.g., *"Wednesday news"*), therefore posing an additional degree of difficulty for a relatedness metric applied on this data.

Native speakers of Spanish, Romanian and Arabic, who were also highly proficient in English, were asked to translate the words in the two data sets. The annotators were provided one word pair at a time, and asked to provide the appropriate translation for each word while taking into account their relatedness within the word pair. The relatedness was meant as a hint to disambiguate the words, when multiple translations were possible.

The annotators were also instructed not to use multi-word expressions in their translations. They were also allowed to use replacement words to overcome slang or culturally-biased terms. For example, in the case of the word pair *dollar-buck*, annotators were allowed to use دينَار[2] as a translation for *buck*.

To test the ability of the bilingual judges to provide correct translations by using this annotation setting, we carried out the following experiment. We collected Spanish translations from five different human judges, which were then merged into a single selection based on the annotators' translation agreement; the merge was done by a sixth human judge, who also played the role of adjudicator when no agreement was reached between the initial annotators.

Subsequently, five additional human experts rescored the word-pair Spanish translations by using the same scale that was used in the construction of the English data set. The correlation between the relatedness scores assigned during this experiment and the scores assigned in the original English experiment was 0.86, indicating that the translations provided by the bilingual judges were correct and preserved the word relatedness.

For the translations provided by the five human judges, in more than 74% of the cases at least three human judges agreed on the same translation for a word pair. When the judges did not provide identical translations, they typically used a close synonym. The high agreement between their translations indicates that the annotation setting was effective in pinpointing the correct translation for each word, even in the case of ambiguous words.

Motivated by the validation of the annotation setting obtained for Spanish, we used only one human annotator to collect the translations for Arabic and Romanian. Table 6 shows examples of translations in the three languages for three word pairs from our data sets.

Using these translations, we create six cross-lingual data sets, one for each possible language pair (English-Spanish, English-Arabic, English-Romanian, Spanish-Arabic, Spanish-Romanian, Arabic-Romanian). Given a source-target language pair, a data set is created by first using the source language for the first word and the target language for the second word, and then reversing the order, i.e., using the source language for the second word and the target language for the first word. The size of the data sets is thus doubled in this way (e.g., the 30 word pairs in the English Miller-Charles set are transformed into 60 word pairs in the English-Spanish Miller-Charles set).

5.2 Results

We evaluate the cross-lingual measure of relatedness on each of the six language pairs. For comparison purposes, we also evaluate the monolingual relatedness on the four languages.

For the evaluation, we use the Pearson (r) and Spearman (ρ) correlation coefficients, which are the standard metrics used in the past for the evaluation of semantic relatedness (Finkelstein et

[2] Arabic for dinars – the commonly used currency in the Middle East.

al., 2001; Zesch et al., 2008; Gabrilovich and Markovitch, 2007). While the Pearson correlation is highly dependent on the linear relationship between the distributions in question, Spearman mainly emphasizes the ability of the distributions to maintain their relative ranking.

Tables 7 and 8 show the results of the evaluations of the cross-lingual relatedness, when using an ESA concept vector with a size of maximum 10,000 concepts.[3]

	English	Spanish	Arabic	Romanian
Miller-Charles				
English	0.58	0.43	0.32	0.50
Spanish		0.44	0.20	0.38
Arabic			0.36	0.32
Romanian				0.58
WordSimilarity-353				
English	0.55	0.32	0.31	0.29
Spanish		0.45	0.32	0.28
Arabic			0.28	0.25
Romanian				0.30

Table 7: Pearson correlation for cross-lingual relatedness on the Miller-Charles and WordSimilarity-353 data sets

	English	Spanish	Arabic	Romanian
Miller-Charles				
English	0.75	0.56	0.27	0.55
Spanish		0.64	0.17	0.32
Arabic			0.33	0.21
Romanian				0.61
WordSimilarity-353				
English	0.71	0.55	0.35	0.38
Spanish		0.50	0.29	0.30
Arabic			0.26	0.20
Romanian				0.28

Table 8: Spearman correlation for cross-lingual relatedness on the Miller-Charles and WordSimilarity-353 data sets

As a validation of our ESA implementation, we compared the results obtained for the monolingual English relatedness with other results reported in the past for the same data sets. Gabrilovich and Markovitch (2007) reported a Spearman correlation of 0.72 for the Miller-Charles data set and 0.75 for the WordSimilarity-353 data set, respectively. Zesch et al. (2008) reported a Spearman correlation of 0.67 for the Miller-Charles set. These values are comparable to the Spearman correlation scores obtained in our experiments for the English data sets (see Table 8), with a fairly large improvement obtained on the Miller-Charles data set when using our implementation.

6 Discussion

Overall, our method succeeds in capturing the cross-lingual semantic relatedness between words. As a point of comparison, one can use the monolingual measures of relatedness as reflected by the diagonals in Tables 7 and 8.

Looking at the monolingual evaluations, the results seem to be correlated with the Wikipedia size for the corresponding language, with the English measure scoring the highest. These results are not surprising, given the direct relation between the Wikipedia size and the sparseness of the ESA concept vectors. A similar trend is observed for the cross-lingual relatedness, with higher results obtained for the languages with large Wikipedia versions (e.g., English-Spanish), and lower results for the languages with a smaller size Wikipedia (e.g., Arabic-Spanish).

For comparison, we ran two additional experiments. In the first experiment, we compared the coverage of our cross-lingual relatedness method to a direct use of the translation links available in Wikipedia. The cross-lingual relatedness is turned into a monolingual relatedness by using the interlanguage Wikipedia links to translate the first of the two words in a cross-lingual pair into the language of the second word in the pair.[4] From the total of 433 word pairs available in the two data sets, this method can produce translations for an average of 103 word pairs per language pair. This means that the direct Wikipedia interlanguage links allow the cross-lingual relatedness measure to be transformed into a monolingual relatedness in about 24% of the cases, which is a low coverage compared to the full coverage that can be obtained with our cross-lingual method of relatedness.

In an attempt to raise the coverage of the translation, we ran a second experiment where we used a state-of-the-art translation engine to translate the first word in a pair into the language of the sec-

[3]The concepts are selected in reversed order of their weight inside the vector in the respective language. Note that the cross-lingual mapping between the concepts in the ESA vectors is done after the selection of the top 10,000 concepts in each language.

[4]We use all the interlanguage links obtained by combining the initial and the discovered links, as described in Section 4.

ond word in the pair. We use Google Translate, which is a statistical machine translation engine that relies on large parallel corpora, to find the most likely translation for a given word. Unlike the previous experiment, this time we can achieve full translation coverage, and thus we are able to produce data sets of equal size that can be used for a comparison between relatedness measures. Specifically, using the translation produced by the machine translation engine for the first word in a pair, we calculate the relatedness within the space of the language of the second word using a monolingual ESA also based on Wikipedia. The results obtained with this method are compared against the results obtained with our cross-lingual ESA relatedness.

Using a Pearson correlation, our cross-lingual relatedness method achieves an average score across all six language pairs of 0.36 for the Miller-Charles data set and 0.30 for the WordSimilarity-353 data set,[5] which is higher than the 0.33 and 0.28 scores achieved for the same data sets when using a translation obtained with Google Translate followed by a monolingual measure of relatedness. These results are encouraging, also given that the translation-based method is limited to those language pairs for which a translation engine exists (e.g., Google Translate covers 40 languages), whereas our method can be applied to any language pair from the set of 250 languages for which a Wikipedia version exists.

To gain further insights, we also determined the impact of the vector length in the ESA concept vector representation, by calculating the Pearson correlation for vectors of different lengths. Figures 1 and 2 show the Pearson score as a function of the vector length for the Miller-Charles and WordSimilarity-353 data sets. The plots show that the cross-lingual measure of relatedness is not significantly affected by the reduction or increase of the vector length. Thus, the use of vectors of length 10,000 (as used in most of our experiments) appears as a reasonable tradeoff between accuracy and performance.

Furthermore, by comparing the performance of the proposed Lesk-like model to the traditional cosine-similarity (Figures 3 and 4), we note that the Lesk-like model outperforms the cosine model on most language pairs. We believe that this is due to the stricter correlation conditions imposed by the cosine-metric in such sparse vector-based representations, as compared to the more relaxed hypothesis used by the Lesk model.

[5]This average considers all the cross-lingual relatedness scores listed in Table 7; it does not include the monolingual scores listed on the table diagonal.

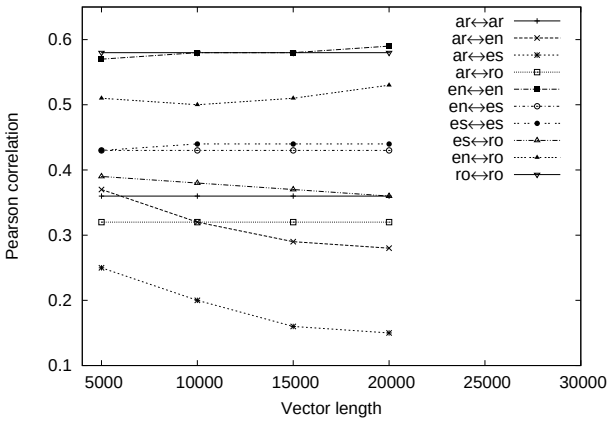


Figure 1: Pearson correlation vs. ESA vector length on the Miller-Charles data set

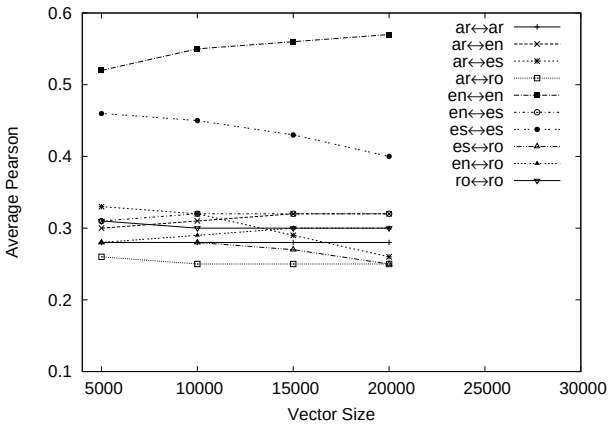


Figure 2: Pearson correlation vs. ESA vector length on the WordSimilarity-353 data set

Finally, we also looked at the relation between the number of interlanguage links found for the concepts in a vector and the length of the vector. Figures 5 and 6 display the average number of interlanguage links as a function of the concept vector length.

By analyzing the effect of the average number of interlanguage links found per word in the given datasets (Figures 5 and 6), we notice that these links increase proportionally with the vector size, as expected. However, this increase does not lead to any significant improvements in accuracy (Figures 1 and 2). This implies that while the presence of interlanguage links is a prerequisite for the mea-

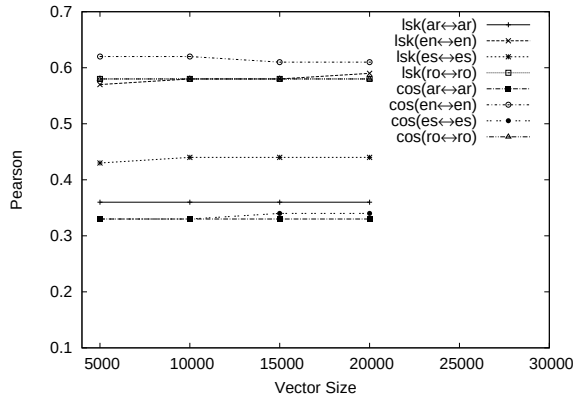


Figure 3: Lesk vs. cosine similarity for the Miller-Charles data set

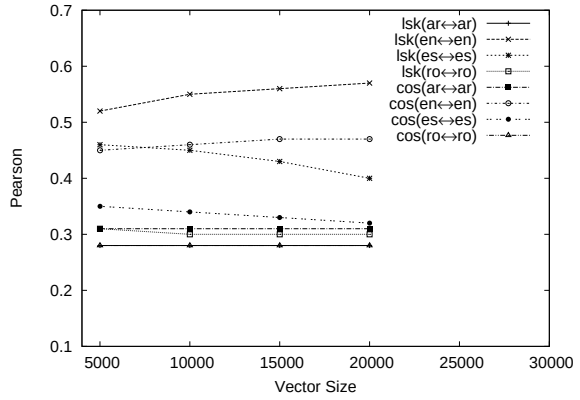


Figure 4: Lesk vs. cosine similarity for the WordSimilarity-353 data set

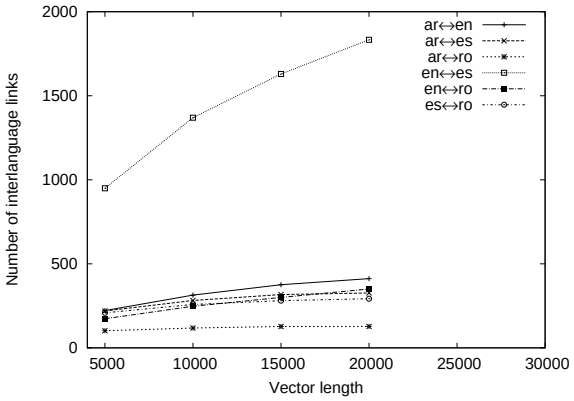


Figure 5: Number of interlanguage links vs. vector length for the Miller-Charles data set

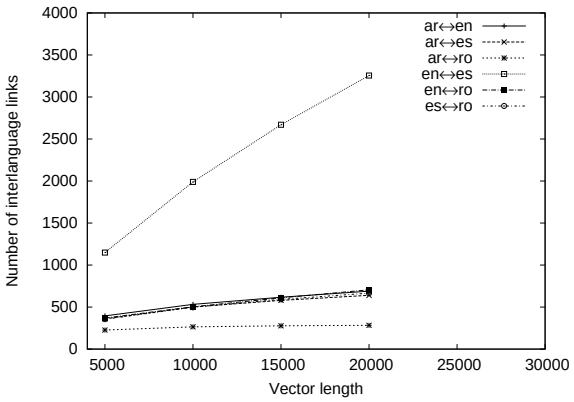


Figure 6: Number of interlanguage links vs. vector length for the WordSimilarity-353 data set

sure of relatedness,[6] their effect is only significant for the top ranked concepts in a vector. Therefore, increasing the vectors size to maximize the matching of the projected dimensions does not necessarily lead to accuracy improvements.

7 Related Work

Measures of word relatedness were found useful in a large number of natural language processing applications, including word sense disambiguation (Patwardhan et al., 2003), synonym identification (Turney, 2001), automated essay scoring (Foltz et al., 1999), malapropism detection (Budanitsky and Hirst, 2001), coreference resolution (Strube and Ponzetto, 2006), and others. Most of the work to date has focused on measures of word relatedness for English, by using methods applied on knowledge bases (Lesk, 1986; Wu and Palmer, 1994; Resnik, 1995; Jiang and Conrath, 1997; Hughes and Ramage, 2007) or on large corpora (Salton et al., 1997; Landauer et al., 1998; Turney, 2001; Gabrilovich and Markovitch, 2007).

Although to a lesser extent, measures of word relatedness have also been applied on other languages, including German (Zesch et al., 2007; Zesch et al., 2008; Mohammad et al., 2007), Chinese (Wang et al., 2008), Dutch (Heylen et al., 2008) and others. Moreover, assuming resources similar to those available for English, e.g., WordNet structures or large corpora, the measures of relatedness developed for English can be in principle applied to other languages as well.

All these methods proposed in the past have been concerned with *monolingual* word relatedness calculated within the boundaries of one language, as opposed to *cross-lingual* relatedness, which is the focus of our work.

The research area closest to the task of cross-

[6] Two languages with no interlanguage links between them will lead to a relatedness score of zero for any word pair across these languages, no matter how strongly related the words are.

lingual relatedness is perhaps cross-language information retrieval, which is concerned with matching queries posed in one language to document collections in a second language. Note however that most of the approaches to date for cross-language information retrieval have been based on direct translations obtained for words in the query or in the documents, by using bilingual dictionaries (Monz and Dorr, 2005) or parallel corpora (Nie et al., 1999). Such explicit translations can identify a direct correspondence between words in two languages (e.g., they will find that *fabbrica* (It.) and *factory* (En.) are translations of each other), but will not capture similarities of a different degree (e.g., they will not find that *lavoratore* (It.; worker in En.) is similar to *factory* (En.).

Also related are the areas of word alignment for machine translation (Och and Ney, 2000), induction of translation lexicons (Schafer and Yarowsky, 2002), and cross-language annotation projections to a second language (Riloff et al., 2002; Hwa et al., 2002; Mohammad et al., 2007). As with cross-language information retrieval, these areas have primarily considered direct translations between words, rather than an entire spectrum of relatedness, as we do in our work.

8 Conclusions

In this paper, we addressed the problem of cross-lingual semantic relatedness, which is a core task for a number of applications, including cross-language information retrieval, cross-language text classification, lexical choice for machine translation, cross-language projections of resources and annotations, and others.

We introduced a method based on concept vectors built from Wikipedia, which are mapped across the interlanguage links available between Wikipedia versions in multiple languages. Experiments performed on six language pairs, connecting English, Spanish, Arabic and Romanian, showed that the method is effective at capturing the cross-lingual relatedness of words. The method was shown to be competitive when compared to methods based on a translation using the direct Wikipedia links or using a statistical translation engine. Moreover, our method has wide applicability across languages, as it can be used for any language pair from the set of 250 languages for which a Wikipedia version exists.

The cross-lingual data sets introduced in this paper can be downloaded from http://lit.csci.unt.edu/index.php/Downloads.

Acknowledgments

The authors are grateful to Carmen Banea for her help with the construction of the data sets. This material is based in part upon work supported by the National Science Foundation CAREER award #0747340. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

S. Bangalore, P. Haffner, and S. Kanthak. 2007. Statistical machine translation through global lexical selection and sentence reconstruction. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic.

A. Budanitsky and G. Hirst. 2001. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Proceedings of the NAACL Workshop on WordNet and Other Lexical Resources*, Pittsburgh.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. 2001. Placing search in context: the concept revisited. In *WWW*, pages 406–414.

P. Foltz, D. Laham, and T. Landauer. 1999. Automated essay scoring: Applications to educational technology. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*, Chesapeake, Virginia.

E. Gabrilovich and S. Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1606–1611.

A. Gliozzo and C. Strapparava. 2006. Exploiting comparable corpora and bilingual dictionaries for cross-language text categorization. In *Proceedings of the Conference of the Association for Computational Linguistics*, Sydney, Australia.

K. Heylen, Y. Peirsman, D. Geeraerts, and D. Speelman. 2008. Modelling word similarity: an evaluation of automatic synonymy extraction algorithms. In *Proceedings of the Sixth International Language Resources and Evaluation*, Marrakech, Morocco.

T. Hughes and D. Ramage. 2007. Lexical semantic knowledge with random graph walks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Prague, Czech Republic.

R. Hwa, P. Resnik, A. Weinberg, and O. Kolak. 2002. Evaluating translational correspondence using annotation projection. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, Philadelphia, July.

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

T. K. Landauer, P. Foltz, and D. Laham. 1998. Introduction to latent semantic analysis. *Discourse Processes*, 25.

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

G. Miller and W. Charles. 1998. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1).

D. Milne. 2007. Computing semantic relatedness using wikipedia link structure. In European Language Resources Association (ELRA), editor, *In Proceedings of the New Zealand Computer Science Research Student Conference (NZCSRSC 2007)*, New Zealand.

S. Mohammad, I. Gurevych, G. Hirst, and T. Zesch. 2007. Cross-lingual distributional profiles of concepts for measuring semantic distance. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL-2007)*, Prague, Czech Republic.

C. Monz and B.J. Dorr. 2005. Iterative translation disambiguation for cross-language information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Salvador, Brazil.

J.-Y. Nie, M. Simard, P. Isabelle, and R. Durand. 1999. Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the Web. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*.

F. Och and H. Ney. 2000. A comparison of alignment models for statistical machine translation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbrucken, Germany, August.

S. Patwardhan, S. Banerjee, and T. Pedersen. 2003. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico City, February.

P. Resnik. 1995. Using information content to evaluate semantic similarity. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, Canada.

E. Riloff, C. Schafer, and D. Yarowsky. 2002. Inducing information extraction systems for new languages via cross-language projection. In *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan, August.

G. Salton, A. Wong, and C.S. Yang. 1997. A vector space model for automatic indexing. In *Readings in Information Retrieval*, pages 273–280. Morgan Kaufmann Publishers, San Francisco, CA.

C. Schafer and D. Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *Proceedings of the 6th Conference on Natural Language Learning (CoNLL 2003)*, Taipei, Taiwan, August.

M. Strube and S. P. Ponzetto. 2006. Wikirelate! computing semantic relatedeness using Wikipedia. In *Proceedings of the American Association for Artificial Intelligence*, Boston, MA.

P. Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001)*, Freiburg, Germany.

X. Wang, S. Ju, and S. Wu. 2008. A survey of chinese text similarity computation. In *Proceedings of the Asia Information Retrieval Symposium*, Harbin, China.

Z. Wu and M. Palmer. 1994. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Mexico.

T. Zesch, I. Gurevych, and M. Mühlhäuser. 2007. Comparing Wikipedia and German Wordnet by Evaluating Semantic Relatedness on Multiple Datasets. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

T. Zesch, C. Müller, and I. Gurevych. 2008. Using Wiktionary for Computing Semantic Relatedness. In *Proceedings of the American Association for Artificial Intelligence*, Chicago.

Joint Optimization for Machine Translation System Combination

Xiaodong He
Microsoft Research
One Microsoft Way, Redmond, WA
xiaohe@microsoft.com

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

Abstract

System combination has emerged as a powerful method for machine translation (MT). This paper pursues a joint optimization strategy for combining outputs from multiple MT systems, where word alignment, ordering, and lexical selection decisions are made jointly according to a set of feature functions combined in a single log-linear model. The decoding algorithm is described in detail and a set of new features that support this joint decoding approach is proposed. The approach is evaluated in comparison to state-of-the-art confusion-network-based system combination methods using equivalent features and shown to outperform them significantly.

1 Introduction

System combination for machine translation (MT) has emerged as a powerful method of combining the strengths of multiple MT systems and achieving results which surpass those of each individual system (e.g. Bangalore, et. al., 2001, Matusov, et. al., 2006, Rosti, et. al., 2007a). Most state-of-the-art system combination methods are based on constructing a confusion network (CN) from several input translation hypotheses, and choosing the best output from the CN based on several scoring functions (e.g. Rosti et. al., 2007a, He et. al., 2008, Matusov et al. 2008). Confusion networks allow word-level system combination, which was shown to outperform sentence re-ranking methods and phrase-level combination (Rosti, et. al. 2007a).

We will review confusion-network-based system combination with the help of the examples in Figures 1 and 2. Figure 1 shows translation hypotheses from three Chinese-to-English MT systems. The general idea is to combine hypotheses in a representation such as the ones in Figure 2, where for each word position there is a set of possible words, shown in columns.[1] The final output is determined by choosing one word from each column, which can be a real word or the empty word ε. For example, the CN in Figure 2a) can generate eight distinct sequences of words, including e.g. "she bought the Jeep" and "she bought the SUV Jeep". The choice is performed to maximize a scoring function using a set of features and a log-linear model (Matusov, et. al 2006, Rosti, et al. 2007a).

We can view a confusion network as an ordered sequence of columns (*correspondence sets*). Each word from each input hypothesis belongs to exactly one correspondence set. Each correspondence set contains at most one word from each input hypothesis and contributes exactly one of its words (including the possible ε) to the final output. Final words are output in the order of correspondence sets. In order to construct such a representation, we need to solve the following two sub-problems: arrange words from all input hypotheses into correspondence sets (alignment problem) and order correspondence sets (ordering problem). After constructing the confusion network we need to solve a third sub-problem: decide which words to output from each correspondence set (lexical choice problem).

In current state-of-the-art approaches, the construction of the confusion network is performed as follows: first, a backbone hypothesis is selected. The backbone hypothesis determines the order of words in the final system output, and guides word-level alignments for construction of columns of possible words at each position. Let us assume that for our example in Figure 1, the second hypothesis is selected as a backbone. All other hypotheses are aligned to the backbone such that these alignments are one-to-one; empty words are inserted where necessary to make one-to-one

[1] This representation is alternative to directed acyclic graph representations of confusion networks.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1202–1211,
Singapore, 6-7 August 2009.

alignment possible. Words in all hypotheses are sorted by the position of the backbone word they align to and the confusion network is determined.

It is clear that the quality of selection of the backbone and alignments has a large impact on the performance, because the word order is determined by the backbone, and the set of possible words at each position is determined by alignment. Since the space of possible alignments is extremely large, approximate and heuristic techniques have been employed to derive them. In pair-wise alignment, each hypothesis is aligned to the backbone in turn, with separate processing to combine the multiple alignments. Several models have been used for pair-wise alignment, starting with TER and proceeding with more sophisticated techniques such as HMM models, ITG, and IHMM (Rosti et. al 2007a, Matusov et al 2008, Krakos et al. 2008, He et al. 2008). A major problem with such methods is that each hypothesis is aligned to the backbone independently, leading to sub-optimal behavior. For example, suppose that we use a state-of-the-art word alignment model for pairs of hypotheses, such as the IHMM. Figure 1 shows likely alignment links between every pair of hypotheses. If Hypothesis 1 is aligned to Hypothesis 2 (the backbone), *Jeep* is likely to align to *SUV* because they express similar Chinese content. Hypothesis 3 is separately aligned to the backbone and since the alignment is constrained to be one-to-one, *SUV* is aligned to *SUV* and *Jeep* to an empty word which is inserted after *SUV*. The network in Figure 2a) is the result of this process. An undesirable property of this CN is that the two instances of Jeep are placed in separate columns and cannot vote to reinforce each other.

Incremental alignment methods have been proposed to relax the independence assumption of pair-wise alignment (Rosti et al. 2008, Li et al. 2009). Such methods align hypotheses to a partially constructed CN in some order. For example, if in such method, Hypothesis 3 is first aligned to the backbone, followed by Hypothesis 1, we are likely to arrive at the CN in Figure 2b) in which the two instances of Jeep are aligned. However, if Hypothesis 1 is aligned to the backbone first, we would still get the CN in Figure 2a). Notice that the desirable output "*She bought the Jeep SUV*" cannot be generated from either of the confusion networks because a re-reordering of columns would be required.

A common characteristic of CN-based approaches is that the order of words (backbone) and the alignment of words (correspondence sets) are decided as greedy steps independently of the lexical choice for the final output. The backbone and alignment are optimized according to auxiliary scoring functions and heuristics which may or may not be optimal with respect to producing CNs leading to good translations. In some recent approaches, these assumptions are relaxed to allow each input hypothesis as a backbone. Each backbone produces a separate CN and the decision of which CN to choose is taken at a later decoding stage, but this still restricts the possible orders and alignments greatly (Rosti et al. 2008, Matusov et al. 2008).

In this paper, we present a joint optimization method for system combination. In this method, the alignment, ordering and lexical selection sub-problems are solved jointly in a single decoding framework based on a log-linear model.

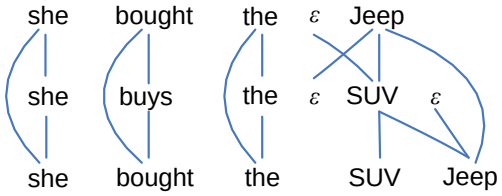


Figure 1. Three MT system hypotheses with pair-wise alignments.

she	bought	the	Jeep	ε
she	buys	the	SUV	ε
she	bought	the	SUV	Jeep

a) Confusion network with pair-wise alignment.

she	bought	the	ε	Jeep
she	buys	the	SUV	ε
she	bought	the	SUV	Jeep

b) Confusion network with incremental alignment.

Figure 2. Correspondence sets of confusion networks under pair-wise and incremental alignment, using the second hypothesis as a backbone.

2 Related Work

There has been a large body of work on MT system combination. Among confusion-network-based algorithms, most relevant to our work are state-of-the-art methods for constructing word alignments (correspondence sets) and methods for improving the selection of a backbone hypothesis. We have already reviewed such work in the introduction and will note relation to

specific models throughout the paper as we discuss specifics of our scoring functions.

In confusion network algorithms which use pair-wise (or incremental) word-level alignment algorithms for correspondence set construction, problems of converting many-to-many alignments and handling multiple insertions and deletions need to be addressed. Prior work has used a number of heuristics to deal with these problems (Matusov, et. al., 2006, He et al 08). Some work has made such decisions in a more principled fashion by computing model-based scores (Matusov et al. 2008), but still special-purpose algorithms and heuristics are needed and a single alignment is fixed.

In our approach, no heuristics are used to convert alignments and no concept of a backbone is used. Instead, the globally highest scoring combination of alignment, order, and lexical choice is selected (subject to search error).

Other than confusion-network-based algorithms, work most closely related to ours is the method of MT system combination proposed in (Jayaraman and Lavie 2005), which we will refer to as J&L. Like our method, this approach performs word-level system combination and is not limited to following the word order of a single backbone hypothesis; it also allows more flexibility in the selection of correspondence sets during decoding, compared to a confusion-network-based approach. Even though their algorithm and ours are broadly similar, there are several important differences.

Firstly, the J&L approach is based on pair-wise alignments between words in different hypotheses, which are hard and do not have associated probabilities. Every word in every hypothesis is aligned to at most one word from each of the remaining hypotheses. Thus there is no uncertainty about which words should belong to the correspondence set of an aligned word *w*, once that word is selected to extend a partial hypothesis during search. If words do not have corresponding matching words in some hypotheses, heuristic matching to currently unused words is attempted.

In contrast, our algorithm is based on the definition of a joint scoring model, which takes into account alignment uncertainty and combines information from word-level alignment models, ordering and lexical selection models, to address the three sub-problems of word-level system combination. In addition to the language model and word-voting features used by the J&L model, we incorporate features which measure alignment confidence via word-level alignment models and features which evaluate re-ordering via distortion models with respect to original hypotheses. While the J&L search algorithm incorporates a number of special-purpose heuristics to address phenomena of unused words lagging behind the last used words, the goal in our work is to minimize heuristics and perform search to jointly optimize the assignment of hidden variables (ordered correspondence sets) and observed output variables (words in final translations).

Finally, the J&L method has not been evaluated in comparison to confusion-network-based methods to study the impact of performing joint decoding for the three sub-problems.

3 Notation

Before elaborating the models and decoding algorithms, we first clarify the notation that will be used in the paper.

We denote by $\boldsymbol{H} = \{h_1, \ldots, h_N\}$ the set of hypotheses from multiple MT systems, where h_i is the hypothesis from the *i*-th system and h_i is a word sequence $w_{i,1}, \ldots, w_{i,L(i)}$ with length $L(i)$. For simplicity, we assume that each system contributes only its 1-best hypothesis for combination. Accordingly, the *i*-th hypothesis h_i will be associated with a weight $W(i)$ which is the weight of the *i*-th system. In the scenario that N-best lists are available from individual systems for combination, the weight of each hypothesis can be computed based on its rank in the N-best list (Rosti et. al. 2007a).

Like in CN-based system combination, we construct a set of ordered correspondence sets (CS) from input hypotheses, and select one word from each CS to form the final output. A CS is defined as a set of (possibly empty) words, one from each hypothesis, that implicitly align to each other and that contributes exactly one of its words to the final output. A valid complete set of CS includes each non-empty word from each hypothesis in exactly one CS. As opposed to CN-based algorithms, our ordered correspondence sets are constructed during a joint decoding process which performs lexical selection at the same time.

To facilitate the presentation of our features, we define notation for ordered CS. A sequence of correspondence sets is denoted by $C = CS_1, \ldots, CS_m$. Each correspondence set is specified by listing the positions of each of the words in the CS in their respective input

hypotheses. Each input hypothesis is assumed to have one special empty word ε at position 0. A CS is denoted by $CS(l_1, \dots, l_N) = \{w_{1,l_1}, \dots, w_{N,l_N}\}$, where w_{i,l_i} is the l_i-th word in the i-th hypothesis and the word position vector $v = [l_1, \dots, l_N]^T$ specifies the position of each word in its original hypothesis. Correspondingly, word w_{i,l_i} has the same weight $W(i)$ as its original hypothesis h_i. As an example, the last two correspondence sets specified by the CN in Figure 2a) would be specified as $CS_4 = CS(4,4,4) = \{Jeep, SUV, SUV\}$ and $CS_5 = CS(0,0,5) = \{\varepsilon, \varepsilon, Jeep\}$.

As opposed to the CS defined in a conventional CN, words that have the same surface form but come from different hypotheses are not collapsed to be one single candidate since they have different original word positions. We need to trace each of them separately during the decoding process.

4 A Joint Optimization Framework For System Combination

The joint decoding framework chooses optimal output according to the following log-linear model:

$$w^* = \underset{w \in \boldsymbol{W}, O \in \boldsymbol{O}, C \in \boldsymbol{C}}{\operatorname{argmax}} \; exp\left\{\sum_{i=1}^{F} \alpha_i \cdot f_i(w, O, C, \boldsymbol{H})\right\}$$

where we denote by $\boldsymbol{C}$ the set of all possible valid arrangements of CS, $\boldsymbol{O}$ the set of all possible orders of CS, $\boldsymbol{W}$ the set of all possible word sequences, consisting of words from the input hypotheses. $\{f_i(w, O, C, \boldsymbol{H})\}$ are the features and $\{\alpha_i\}$ are the feature weights in the log-linear model, respectively.

4.1 Features

A set of features are used in this framework. Each of them models one or more of the alignment, ordering, and lexical selection sub-problems. Features are defined as follows.

Word posterior model:

The word posterior feature is the same as the one proposed by Rosti et. al. (2007a). i.e.,

$$f_{wp}(w, O, C, \boldsymbol{H}) = \sum_{m=1}^{M} log\big(P(w_m | CS_m)\big)$$

where the posterior of a single word in a CS is computed based on a weighted voting score:

$$P\big(w_{i,l_i} | CS\big) = P\left(w_{i,l_i} \middle| CS(l_1, \dots, l_N)\right) = \sum_{k=1}^{N} W(k)\, \delta(w_{k,l_k} = w_{i,l_i})$$

and M is the number of CS generated. Note that M may be larger than the length of the output word sequence w since some CS may generate empty words.

Bi-gram voting model:

The second feature we used is a bi-gram voting feature proposed by Zhao and He (2009), i.e., for each bi-gram $\langle w_i, w_{i+1} \rangle$, a weighted position-independent voting score is computed:

$$P(\langle w_i, w_{i+1} \rangle | \boldsymbol{H}) = \sum_{k=1}^{N} W(k)\, \delta(\langle w_i, w_{i+1} \rangle \in h_i)$$

And the global bi-gram voting feature is defined as:

$$f_{bgv}(w, O, C, \boldsymbol{H}) = \sum_{i=1}^{|w|-1} log\big(P(\langle w_i, w_{i+1} \rangle | \boldsymbol{H})\big)$$

Distortion model:

Unlike in the conventional CN-based system combination, flexible orders of CS are allowed in this joint decoding framework. In order to model the distortion of different orderings, a distortion model between two CS is defined as follows:

First we define the distortion cost between two words at a single hypothesis. Similarly to the distortion penalty in the conventional phrase-based decoder (Koehn 2004b), the distortion cost of jumping from a word at position i to another word at position j, $d(i,j)$, is proportional to the distance between i and j, e.g., $|i-j|$. Then, the distortion cost of jumping from one CS, which has a position vector recording the original position of each word in that CS, to another CS is a weighted sum of single-hypothesis-based distortion costs:

$$d(CS_m, CS_{m+1}) = \sum_{k=1}^{N} W(k) \cdot |l_{m,k} - l_{m+1,k}|$$

where $l_{m,k}$ and $l_{m+1,k}$ are the k-th element of the word position vector of CS_m and CS_{m+1}, respectively. For the purpose of computing the distortion feature, the position of an empty word is taken to be the same as the position of

the last visited non-empty word from the same hypothesis.

The overall ordering feature can then be computed based on $d(CS_m, CS_{m+1})$:

$$f_{dis}(w, O, C, \boldsymbol{H}) = -\sum_{m=1}^{M-1} d(CS_m, CS_{m+1})$$

It is worth noting that this is not the only feature modeling the re-ordering behavior. Under the joint decoding framework, other features such as the language model and bi-gram voting affect the ordering as well.

Alignment model:

Each CS consists of a set of words, one from each hypothesis, that are implicitly aligned to each other. Therefore, a valid complete set of CS defines the word alignment among different hypotheses. In this paper, we derive an alignment score of a CS based on alignment scores of word pairs in that CS. To compute scores for word pairs, we perform pair-wise hypothesis alignment using the indirect HMM (He et al. 2008) for every pair of input hypotheses. Note that this involves a total of *N* by (*N*-1)/2 bi-directional hypothesis alignments. The alignment score for a pair of words w_{j,l_j} and w_{k,l_k} is defined as the average of posterior probabilities of alignment links in both directions and is thus direction independent:

$$p\left(w_{j,l_j}, w_{k,l_k}\right) = \frac{1}{2}\left(p(a_{l_j} = l_k | h_j, h_k) + p(a_{l_k} = l_j | h_k, h_j)\right)$$

If one of the two words is ε, the posterior of aligning word ε to state j is computed as suggested by Liang et al. (2006), i.e.,

$$p(a_0 = l_j | h_k, h_j) = \prod_{i=1}^{L(k)} \left(1 - p(a_i = l_j | h_k, h_j)\right)$$

And $p(a_{l_j} = 0 | h_j, h_k)$ can be computed by the HMM directly.

If both words are ε, then a pre-defined $p_{\varepsilon\varepsilon}$ is assigned, i.e., $p(a_0 = 0 | h_k, h_j) = p_{\varepsilon\varepsilon}$, where $p_{\varepsilon\varepsilon}$ can be optimized on a held-out validation set.

For a CS of words, if we set the *j*-th word as an anchor word, the probability that all other words align to that word is:

$$p(j|CS) = \prod_{\substack{k=1 \\ k \neq j}}^{N} p\left(w_{j,l_j}, w_{k,l_k}\right)$$

The alignment score of the whole CS is a weighted sum of the logarithm of the above alignment probabilities, i.e.,

$$S_{aln}(CS) = \sum_{j=1}^{N} W(j)\, log(P(j|CS))$$

and the global alignment score is computed as:

$$f_{aln}(w, O, C, \boldsymbol{H}) = \sum_{m=1}^{M} S_{aln}(CS_m)$$

Entropy model:

In general, it is preferable to align the same word from different hypotheses into a common CS. Therefore, we use entropy to model the purity of a CS. The entropy of a CS is defined as:

$$Ent(CS) = Ent(CS(l_1, \ldots, l_N)) = \sum_{i=1}^{N'} P(w_{i,l_i}|CS) logP(w_{i,l_i}|CS)$$

where the sum is taken over all distinct words in the CS. Then the global entropy score is computed as:

$$f_{ent}(w, O, C, \boldsymbol{H}) = \sum_{m=1}^{M} Ent(CS_m)$$

Other features used in our log-linear model include the count of real words |*w*|, a n-gram language model, and the count *M* of CS sets.

These features address one or more of the three sub-problems of MT system combination. By performing joint decoding with all these features working together, we hope to derive better decisions on alignment, ordering and lexical selection.

5 Joint Decoding

5.1 Core algorithm

Decoding is based on a beam search algorithm similar to that of the phrase-based MT decoder (Koehn 2004b). The input is a set of translation hypotheses to be combined, and the final output

sentence is generated left to right. Figure 3 illustrates the decoding process, using the example input hypotheses from Figure 1. Each decoding state represents a partial sequence of correspondence sets covering some of the words in the input hypotheses and a sequence of words selected from the CS to form a partial output hypothesis. The initial decoding state has an empty sequence of CS and an empty output sequence. A state corresponds to a complete output candidate if its CS covers all input words.

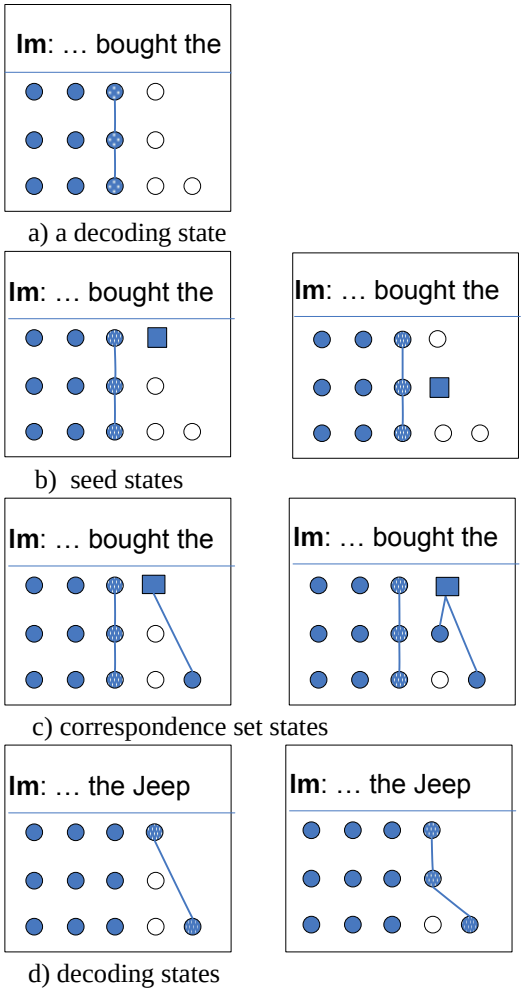


Figure 3. Illustration of the decoding process.

In practice, because the features over hypotheses can be decomposed, we do not need to encode all of this information in a decoding state. It suffices to store a few attributes. They include positions of words from each input hypothesis that have been visited, the last two non-empty words generated (if a tri-gram LM is used), and an "end position vector (EPV)" recording positions of words in the last CS, which were just visited. In the figure, the visited words are shown with filled circles and the EPV is shown with a dotted pattern in the filled circles. Words specified by the EPV are implicitly aligned. In the state in Figure 3 a) the first three words of each hypothesis have been visited, the third word of each hypothesis is the last word visited (in the EPV), and the last two words produced are "*bought the*". The states also record the decoding score accumulated so far and an estimated future score to cover words that have not been visited yet (not shown).

The expansion from one decoding state to a set of new decoding states is illustrated in Figure 3. The expansion is done in three steps with the help of intermediate states. Starting from a decoding state as shown in Figure 3a), first a set of "seed states" as shown in Figure 3b) are generated. Each seed state represents a choice of one of unvisited words, called a "seed word" which is selected and marked as visited. For example, the word *Jeep* from the first hypothesis and the word *SUV* from the second hypothesis are selected in the two seed states shown in Figure 3b), respectively. These seed states further expand into a set of "CS states" as shown in Figure 3c). I.e., a CS is formed by picking one word from each of the other hypotheses which is unvisited and has a valid alignment link to the seed word. Figure 3c) shows two CS states expanded from the first seed state of Figure 3b), using *Jeep* from the first hypothesis as a seed word. In one of them the empty word from the second hypothesis is chosen, and in the other, the word *SUV* is chosen. Both are allowed by the alignments illustrated in Figure 1. Finally, each CS state generates one or more complete decoding states, in which a word is chosen from the current CS and the EPV vector is advanced to reflect the last newly visited words. Figure 3d) shows two such states, descending from the corresponding CS states in 3c). After one more expansion the state in 3d) on the left can generate the translation "*She bought the Jeep SUV*", which cannot be produced by either confusion network in Figure 2.

5.2 Pruning

The full search space of joint decoding is a product of the alignment, ordering, and lexical selection spaces. Its size is exponential in the length of the sentence and the number of hypotheses involved in combination. Therefore, pruning techniques are necessary to reduce the search space.

First we will prune down the alignment space. Instead of allowing any alignment link between

arbitrary words of two hypotheses, only links that have alignment score higher than a threshold are allowed, plus links in the union of the Viterbi alignments in both directions. In order to prevent the garbage collection problem where many words align to a rare word at the other side (Moore, 2004), we further impose the limit that if one word is aligned to more than *T* words, these links are sorted by their alignment score and only the top *T* links are kept. Meanwhile, alignments between a real word and ε are always allowed.

We then prune down the ordering space by limiting the expansion of new states. Only states that are *adjacent* to their preceding states are created. Two states are called *adjacent* if their EPVs are adjacent, i.e., given the EPV of the preceding state m as $[l_{m,1}, \ldots, l_{m,N}]^T$ and the EPV of the next state m+1 as $[l_{m+1,1}, \ldots, l_{m+1,N}]^T$, if at least at one dimension k, $l_{m+1,k} = l_{m,k}+1$, then these two states are *adjacent*. When checking the adjacency of two states, the position of an empty word is taken to be the same as the position of the last visited non-empty word from the same hypothesis.

The number of possible CS states expanded from a decoding state is exponential in the number of hypotheses. In decoding, these CS states are sorted by their alignment scores and only the top *K* CS states are kept.

The search space can be further pruned down by the widely used technique of path recombination and by best-first pruning.

Path recombination is a risk-free pruning method. Two paths can be recombined if they agree on a) words from each hypothesis that have been visited so far, b) the last two real words generated, and c) their EPVs. In such case, we only need to keep the path with the higher score.

Best-first pruning can help to reduce the search space even further. In the decoding process we compare paths that have generated the same number of words (both real and empty words) and only keep a certain number of most promising paths. Pruning is based on an estimated overall score of each path, which is the sum of the decoding score accumulated so far and an estimated future score to cover the words that have not been visited. Next we discuss the future score computation.

5.3 Computing the future score

In order to estimate the future cost of an unfinished path, we treat the unvisited words of one input hypothesis as a backbone, and apply a greedy search for alignment based on it; i.e., for each word of this backbone, the most likely words (based on the alignment link scores) from other hypotheses, one word from each hypothesis, are collected to form a CS. These CS are ordered according to the word order of the backbone and form a CN. Then, a light decoding process with a search beam of size one is applied to decode this CN and find the approximate future path, with future feature scores computed during the decoding process. If there are leftover words not included in this CN, they are treated in the way described in section 5.4. Additionally, caching techniques are applied to speed up the computation of future scores further.

Given the method discussed above, we can estimate a future score based on each input hypothesis, and the final future score is estimated as the best of these hypothesis-dependent scores.

5.4 Dealing with leftover input words

At a certain point a path will reach the end, i.e., no more states can be generated from it according to the state expansion requirement. Then it is marked as a finished path. However, sometimes the state may contain a few input words that have not been visited. An example of this situation is the second state in Figure 3d). The word *SUV* in the third input hypothesis is left unvisited and it cannot be selected next because there is no adjacent state that can be generated. For such cases, we need to compute an extra score of covering these leftover words. Our approach is to create a state that produces the same output translation, but also covers all remaining words. For each leftover word, we create a pseudo CS that contains just that word plus ε's from all other hypotheses, and let it output ε. Moreover, that CS is inserted at a place such that no extra distortion cost is incurred. Figure 4 shows an example using the second state in Figure 3d). The last two words from the first two MT hypotheses "*the Jeep*" and "*the SUV*" align to the third and fifth words of the third hypothesis "*the Jeep*"; the word $w_{3,4}$ from the third hypothesis is left unvisited. The original path has two CS and one left-over word $w_{3,4}$. It is expanded to have three CS, with a pseudo CS inserted between the two CS.

It is worth noting that the new inserted pseudo CS will not affect the word count feature and contextually dependent feature scores such as the LM and bi-gram voting, since it only generates an empty word. Moreover, it will not affect the

distortion score either. For example, as shown in Figure 4, the distortion cost of jumping from word $w_{2,3}$ to ε_2 and then to $w_{2,4}$ is the same as the cost of jumping from $w_{2,3}$ to $w_{2,4}$ given the way we assign position to empty word and the fact that the distortion cost is proportional to the difference between word positions.

Scores of other features for this pseudo CS such as word posterior (of ε), alignment score, CS entropy, and CS count are all local scores and can be computed easily. Unlike future scores which are approximate, the score computed in this process is exact. Adding this extra score to the existing score accumulated in the final state gives the complete score of this finished path. When all paths are finished, the one with the best complete score is returned as the final output sentence.

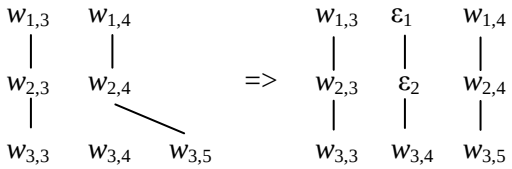


Figure 4. Expanding a leftover word to a pseudo correspondence set.

6 Evaluation

6.1 Experimental conditions

For the joint decoding method, the threshold for alignment-score-based pruning is set to 0.25 and the maximum number of words that can align to the same word is limited to 3. We call this the *standard setting*. The joint decoding approach is evaluated on the Chinese-to-English (C2E) test set of the 2008 NIST Open MT Evaluation (NIST 2008). Results are reported in case insensitive BLEU score in percentages (Papineni et. al., 2002).

The NIST MT08 C2E test set contains 691 and 666 sentences of data from two genres, newswire and web-data, respectively. Each test sentence has four references provided by human translators. Individual systems in our experiments belong to the official submissions of the MT08 C2E constraint-training track. Each submission provides 1-best translation of the whole test set. In order to train feature weights, the original test set is divided into two parts, called the dev and test set, respectively. The dev set consists of the first half of both newswire and web-data, and the test set consists of the second half of data of both genres.

There are 20 individual systems available. We ranked them by their BLEU score results on the dev set and picked the top five systems, excluding systems ranked 5th and 6th since they are subsets of the first entry (NIST 2008). Performance of these systems on the dev and test sets is shown in Table 1.

The baselines include a pair-wise hypothesis alignment approach using the indirect HMM (IHMM) proposed by He et al. (2008), and an incremental hypothesis alignment approach using the incremental HMM (IncHMM) proposed by Li et al. (2009). The lexical translation model used to compute the semantic similarity is estimated from two million parallel sentence-pairs selected from the training corpus of MT08. The backbone for the IHMM-based approach is selected based on Minimum Bayes Risk (MBR) using a BLEU-based loss function. The various parameters of the IHMM and the IncHMM are tuned on the dev set. The same IHMM is used to compute the alignment feature score for the joint decoding approach.

The final combination output can be obtained by decoding the CN with a set of features. The features used for the baseline systems are the same as the features used by the joint decoding approach. Some of these features are constant across decoding hypotheses and can be ignored. The non-constant features are word posterior, bi-gram voting, language model score, and word count. They are computed in the same way as for the joint decoding approach.

System weights and feature weights are trained together using Powell's search for the IHMM-based approach. Then the same system weights are applied to both IncHMM and Joint Decoding -based approaches, and the feature weights of them are trained using the max-BLEU training method proposed by Och (2003) and refined by Moore and Quirk (2008).

Table 1: Performance of individual systems on the dev and test set

System ID	dev	test
System A	32.88	31.81
System B	32.82	32.03
System C	32.16	31.87
System D	31.40	31.32
System E	27.44	27.67

6.2 Comparison against baselines

Table 2 lists the BLEU scores achieved by the two baselines and the joint decoding approach. Both baselines surpass the best individual system

significantly. However, the gain of incremental HMM over IHMM is smaller than that reported in Li et al. (2009). One possible reason of such discrepancy could be that fewer hypotheses are used for combination in this experiment compared to that of Li et al. (2009), so the performance difference between them is narrowed accordingly. Despite that, the proposed joint decoding method outperforms both IHMM and IncHMM baselines significantly.

Table 2: Comparison between the joint decoding approach and the two baselines

method	dev	test
IHMM	36.91	35.85
IncHMM	37.32	36.38
Joint Decoding	37.94	37.20*

* The gains of Joint Decoding over IHMM and IncHMM are both with a statistical significance level > 99%, measured based on the paired bootstrap re-sampling method (Koehn 2004a)

6.3 Comparison of alignment pruning

The effect of alignment pruning is also studied. We tested with limiting the allowable links to just those that in the union of bi-directional Viterbi alignments.

The results are presented in Table 3. Compared to the standard setting, allowing only links in the union of the bi-directional Viterbi alignments causes slight performance degradation. On the other hand, it still outperforms the IHMM baseline by a fair margin. This is because the joint decoding approach is effectively resolving the ambiguous 1-to-many alignments and deciding proper places to insert empty words during decoding.

Table 3: Comparison between different settings of alignment pruning

Setting	Test
standard settings	37.20
union of Viterbi	36.88

6.4 Comparison of ordering constraints

In order to investigate the effect of allowing flexible word ordering, we conducted experiments using different constraints on the ordering of CS in the decoding process. In the first case, we restrict the order of CS to follow the word order of a backbone, which is one of the input hypotheses selected by MBR-BLEU. In the second case, the order of CS is constrained to follow the word order of at least one of the input hypotheses. As shown in Table 4, in comparison to the standard setting that allows backbone-free word ordering, the constrained settings did not lead to significant performance degradation. This indicates that most of the gain due to the joint decoding approach comes from the joint optimization of alignment and word selection. It is possible, though, that if we lift the CS *adjacency* constraint during search, we might derive more benefit from flexible word ordering.

Table 4: Effect of ordering constraints

Setting	test
standard settings	37.20
monotone w.r.t. backbone	37.22
monotone w.r.t. any hyp.	37.12

7 Discussion

This paper proposed a joint optimization approach for word-level combination of translation hypotheses from multiple machine translation systems. Unlike conventional confusion-network-based methods, alignments between words from different hypotheses are not pre-determined and flexible word orderings are allowed. Decisions on word alignment between hypotheses, word ordering, and the lexical choice of the final output are made jointly according to a set of features in the decoding process. A new set of features to model alignment and re-ordering behavior is also proposed. The method is evaluated against state-of-the-art baselines on the NIST MT08 C2E task. The joint decoding approach is shown to outperform baselines significantly.

Because of the complexity of search, a challenge for our approach is combining a large number of input hypotheses. When N-best hypotheses from the same system are added, it is possible to pre-compute and fix the one-to-one word alignment among the same-system hypotheses; such pre-computation is reasonable given our observation that the disagreement among hypotheses from different systems is larger than that among hypotheses from the same system. This will reduce the alignment search space to be the same as that for 1-best case. We plan to study this setting in future work.

To further improve the performance of our approach we see the biggest opportunity in developing better estimates of future scores and incorporating additional features. Beside potential performance improvement, they may help on more effective pruning and speed up the overall decoding process as well.

References

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Proceedings of IEEE ASRU*.

Xiaodong He, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore 2008. Indirect HMM based Hypothesis Alignment for Combining Outputs from Machine Translation Systems. In *Proceedings of EMNLP*.

Shyamsundar Jayaraman and Alon Lavie. 2005. Multi-engine machine translation guided by explicit word matching. In *Proceedings of EAMT*.

Damianos Karakos, Jason Eisner, Sanjeev Khudanpur, and Markus Dreyer 2008. Machine Translation System Combination using ITG-based Alignments. In *Proceedings of ACL*.

Philipp Koehn, 2004a, Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of EMNLP*.

Philipp Koehn. 2004b. Pharaoh: A Beam Search Decoder For Phrase Based Statistical Machine Translation Models. In *Proceedings of AMTA*.

Chi-Ho Li, Xiaodong He, Yupeng Liu and Ning Xi, 2009. Incremental HMM Alignment for MT System Combination. In *Proceedings of ACL*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Personal Communication

Evgeny Matusov, Nicola Ueffing and Hermann Ney. 2006. Computing Consensus Translation from Multiple Machine Translation Systems using Enhanced Hypothesis Alignment. In *Proceedings of EACL*.

Evgeny Matusov, Gregor Leusch, Rafael E. Banchs, Nicola Bertoldi, Daniel Déchelotte, Marcello Federico, Muntsin Kolss, Young-Suk Lee, José B. Mariño, Matthias Paulik, Salim Roukos, Holger Schwenk, and Hermann Ney. 2008. System combination for machine translation of spoken and written language. *IEEE transactions on audio speech and language processing* 16(7).

Robert C. Moore and Chris Quirk. 2008. Random Restarts in Minimum Error Rate Training for Statistical Machine Translation, In *Proceedings of COLING*

Robert C. Moore. 2004. Improving IBM Word Alignment Model 1, In *Proceedings of ACL*.

NIST 2008. The NIST Open Machine Translation Evaluation.www.nist.gov/speech/tests/mt/2008/doc/

Franz J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of ACL*.

Kishore Papineni, Salim Roukos, Todd Ward and Wei- Jing Zhu 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*.

Antti-Veikko I. Rosti, Bing Xiang, Spyros Matsoukas, Richard Schwartz, Necip Fazil Ayan, and Bonnie J. Dorr. 2007a. Combining outputs from multiple machine translation systems. In *Proceedings of NAACL-HLT*.

Antti-Veikko I. Rosti, Spyros Matsoukas, and Richard Schwartz 2007b. Improved Word-level System Combination for Machine Translation. In *Proceedings of ACL*.

Antti-Veikko I. Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz 2008. Incremental Hypothesis Alignment for Building Confusion Networks with Application to Machine Translation System Combination. In *Proceedings of the 3rd ACL Workshop on SMT*.

Yong Zhao and Xiaodong He, 2009. Using N-gram based Features for Machine Translation System Combination. In *Proceedings of NAACL-HLT*

Fully Lexicalising CCGbank with Hat Categories

Matthew Honnibal and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{mhonn,james}@it.usyd.edu.au

Abstract

We introduce an extension to CCG that allows form and function to be represented simultaneously, reducing the proliferation of modifier categories seen in standard CCG analyses.

We can then remove the non-combinatory rules CCGbank uses to address this problem, producing a grammar that is fully lexicalised and far less ambiguous.

There are intrinsic benefits to full lexicalisation, such as semantic transparency and simpler domain adaptation. The clearest advantage is a 52-88% improvement in parse speeds, which comes with only a small reduction in accuracy.

1 Introduction

Deep grammars return parses that map transparently to semantic analyses, allowing information extraction systems to deal directly with content representations. Usually, this mapping is lexically specified, by linking lexical entries to semantic analyses. This property, lexicalisation, is central to some of the linguistic theories behind deep grammars, particularly Combinatory Categorial Grammar (Steedman, 2000) and Lexicalised Tree Adjoining Grammar (Joshi, 1999).

Lexicalisation can also help deep grammars achieve satisfactory parse times. Lexicalised grammars use few rules, which simply manipulate the lexical categories. The categories can be quickly assigned in a supertagging pre-process, dramatically reducing the search space the parser must explore (Bangalore and Joshi, 1999).

Combinatory Categorial Grammar (CCG) is well suited to this strategy, and Clark and Curran (2007) have highlighted the division of labour between the parser and the supertagger as one of the critical aspects of their approach to statistical CCG parsing. In their system, the division is managed with parameters that control how many categories the parser's chart is seeded with. But even if the parser is only given one category per word, it still has a lot of freedom — because the grammar it uses is not fully lexicalised.

In a fully lexicalised CCG grammar, modifier categories refer to the category of their head. This category does not necessarily represent the head's constituent type. For instance, the category of an adverb like *still* depends on whether it is modifying a predicate verb (1), or a clausal adjunct (2):

(1)	*The lion*	*was*	*lying*	*still*
	NP	VP/VP	*VP*	$VP\backslash VP$
(2)	*The lion*	*waited,*	*lying*	*still*
	NP	*VP*	$VP\backslash VP$	$(VP\backslash VP)\backslash(VP\backslash VP)$

Analyses like these are problematic because the training data is unlikely to include examples of each word in every syntactic environment that requires a new category. Hockenmaier and Steedman's (2007) solution was to add category specific phrase-structure rules to the grammar, which disrupts the linguistic principles of the formalism, and introduces over-generation and ambiguity as shown in Figure 1.

This paper proposes a new way to balance lexical and grammatical ambiguity in CCG. We introduce an extension to the formalism that allows type-changing rules to be lexically specified. The extension adds a new field to the category objects, and one additional rule to utilise it. This allows the formalism to express type-changing operations in a theoretically desirable way.

Lexically specifying the type-changing rules reduces the ambiguity in the grammar substantially, which leads to substantial improvements in parsing efficiency. After modifying the C&C parser and CCGbank, the parser runs almost twice as quickly, with only a 0.5% reduction in accuracy.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1212–1221,
Singapore, 6-7 August 2009.

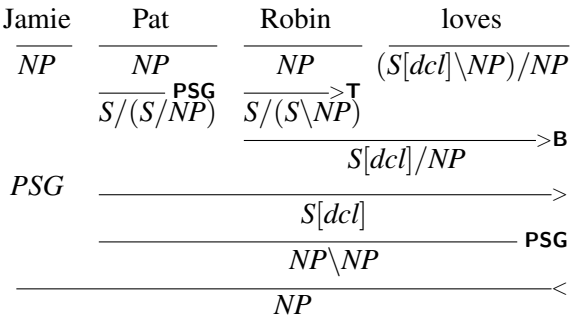


Figure 1: Over-generation by CCGbank rules.

2 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) (Steedman, 2000) is a lexicalised grammar formalism based on categorial grammar (Bar-Hillel, 1953). CCG can be distinguished from other CG extensions, such as categorial type-logic (Moortgat, 1997) by its attention to linguistic minimalism. One aim of the theory is to explain universal constraints on natural language syntax, so the generative power of the formalism is intended to closely match what natural language seems to require.

Steedman and Baldridge (2007) argue that the requirements can be fulfilled almost entirely by two basic rule types: application and composition. Direction specific instances of these types yields a grammar that consists of just six rules.

Initially, it seemed that some of the rules had to be restricted to certain contexts, particularly in languages that did not allow scrambling. Baldridge and Kruijff (2003) have since shown that rules could be restricted lexically, using a hierarchy of slash subtypes. This relieved the need for any language specific meta-rules, allowing CCG to offer a completely universal grammar, and therefore a theory of the innate human language faculty.

With a universal grammar, language specific variation is confined to the lexicon. A CCG lexical category is either an atomic type, like N, or a function that specifies an argument in a particular direction, and a result, like $S\backslash NP$ (where S is the result, NP the argument, and $\backslash$ indicates the argument must be found to the left).

Hockenmaier and Steedman (2007) showed that a CCG corpus could be created by adapting the Penn Treebank (Marcus et al., 1993). CCGbank has since been used to train fast and accurate CCG parsers (Clark and Curran, 2007).

3 The Need for Type-changing in CCG

We argue that there is a clear need for some sort of type-changing mechanism in CCG. The practical need for this has been known since at least Hockenmaier (2003), who introduced a type-changing mechanism into CCGbank in order to control the problem referred to as *modifier category proliferation*. We briefly describe the problem, and then the prominent solutions that have been proposed.

Unlike formalisms like LTAG and HPSG, CCG does not use different grammatical rules for arguments and adjuncts. Instead, modifier categories take the form $X_1|X_1$, where X is the category of the constituent being modified, and the subscript indicates that the result should inherit from the argument via unification. The modifier can then use the application rule to attach to its head, and return the head unchanged:

(3) *unusually* $(S[adj]\backslash NP)/(S[adj]\backslash NP)$ *resilient* $S[adj]\backslash NP$

unusually here modifies the predicative adjective *resilient*, attaching it as an argument using forward application. This prevents *resilient* from having to subcategorise for adjuncts, since they are optional. The problem is that *unusually* must subcategorise for the function of its head. If *resilient* changes function and becomes a noun modifier, its modifiers must change category too:

(4) *an* NP/N *unusually* $(N/N)/(N/N)$ *resilient* N/N *strain* N

There is often a way to analyse around the need for type-changing operations in CCG. However, these solutions tend to cause new difficulties, and the resulting category ambiguity is quite problematic (Hockenmaier and Steedman, 2002). The fact is that form-to-function coercions are quite common in English, so the grammar needs a way to have a constituent be modified according to its form, before undergoing a type-change to its function category.

One way to describe the problem is to say that CCG categories have an over-extended *domain of locality* (Joshi, 1999), the part of the derivation that it describes. A category should specify all and only the dependencies it governs, but CCG modifier categories are often forced to specify their heads' dependencies as well. These undesirable notational dependencies can also prevent modifier categories from factoring recursion away from their domain of locality.

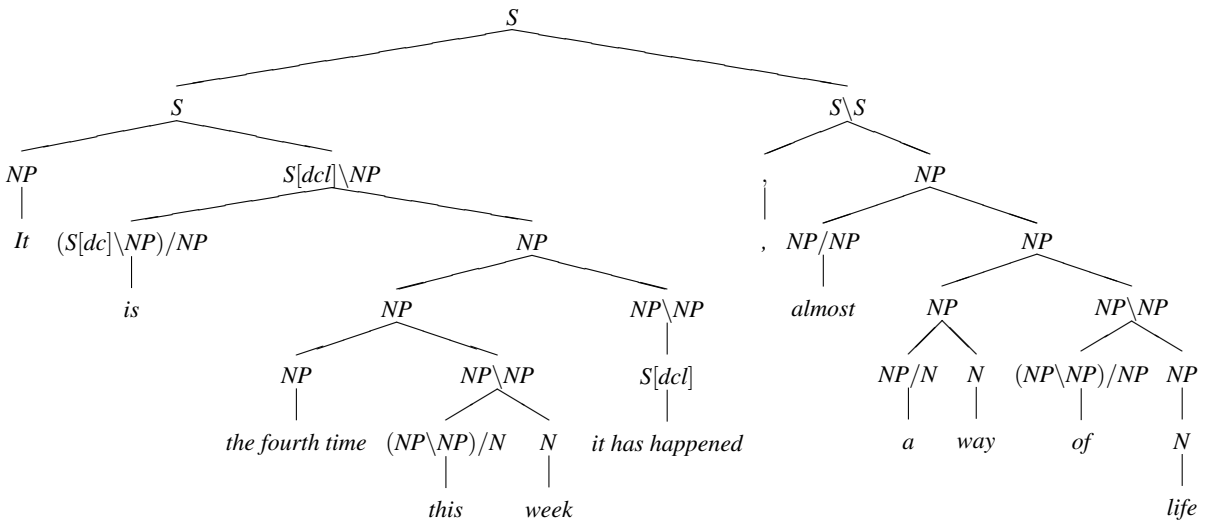


Figure 2: CCGbank derivation showing PSG rules.

4 Problems with Existing Proposals

This section completes the motivation of the paper by arguing that the existing proposals for type-changing are linguistically unsatisfactory, practically difficult, or a combination of the two.

4.1 Problems with PSG Rules

Hockenmaier and Steedman (2002) includes a brief discussion of the modifier category proliferation problem, and introduces unary phrase-structure rules to address the situation. Figure 2 shows two such rules. The $<S[dcl] \rightarrow NP\backslash NP>$[1] rule allows the reduced relative clause, *it has happened*, to be analysed as a modifier without affecting the category any modifiers that might attach to it. The other PSG type-changing rule in the derivation, $<, NP \rightarrow S\backslash S>$ enables the extraposition, using the punctuation to make the rule more precise.

One alternative to type-changing rules here would be to have *time* subcategorise for the clause, with a category like $NP/S[dcl]$. This would capture the constraint that only a restricted subset of nouns can be extracted as adjuncts in this way. The problem is that the extra argument would interfere with the attachment of adjuncts like *this week* to the *NP*, because the $NP\backslash NP$ category cannot be allowed to participate in backwards cross-composition rules (Baldridge and Kruijff, 2003).

There are 204 type-changing PSG rules in the training partition of CCGbank. 53 of the frequent rules transform produce modifier categories, 48 of them transforming verbal categories. The PSG rules also handle a variety of other constructions, such as form/function discrepancies like gerund nominals. By far the most frequent rule, with 115,333 occurrences, is $<N \rightarrow NP>$, which transforms bare nominals into noun phrases.

Steedman and Baldridge (2007) describes the CCG grammar as consisting of just 6 language universal combinatory rules, plus two lexical operations (type raising). Not only do the 204 category specific type-changing rules in CCGbank make the grammar ambiguous, they also run contrary to the design principles of the formalism.

CCG is a linguistically motivated formalism, which means it is not only interested in providing a convenient, computable notation for grammar development. In addition, it constitutes a hypothesis about the nature of the human language faculty. Like other lexicalised formalisms, part of the theory is that it is the grammar that is innate, and the lexicon is acquired.

If the grammar is innate, it must be language universal, confining all language specific variation to the lexicon. Baldridge and Kruijff (2003) described how the remaining language specific grammatical constraints described by Steedman (2000) could be controlled in the lexicon, using multi-modal slashes that have since become integrated into the main body of the theory (Steedman and Baldridge, 2007).

In addition to being linguistically undesirable, the PSG rules in CCGbank produce practical difficulties. Every additional rule increases ambiguity, motivating the C&C system to choose to implement only the most frequent. This decreases

[1] Phrase-structure rules are presented in bottom-up notation.

the parser's coverage, and introduces another dimension of domain sensitivity. For instance, the type-changing rule that allows gerund nominals, $<S[ng]\backslash NP \rightarrow NP>$, occurs roughly 300 times in the training data. The parser does not implement this rule, so if it is ported to a new domain, where the construction is frequent, the rule will have to be added. Presumably, the parser would also benefit from the removal of rules which are infrequent in some new, target domain.

The restricted set of PSG rules the parser does implement results in considerable added ambiguity to the grammar. Figure 1 shows how the rules interact to produce over-generation.

The PSG rules are also a barrier to the semantic transparency of the theory, one of its most attractive properties for natural language engineering. CCG derivations are isomorphic to semantic analyses, because the derivation instantiates dependencies between CCG categories that can be paired with semantic categories. This isomorphism is disrupted by the addition of PSG rules, since the grammar is no longer lexicalised. Often, the rules can be semantically annotated, restoring the isomorphism; but sometimes, this cannot be done.

For instance, the extraposition rule in Figure 2 transforms the *NP* category into $S\backslash S$. There is no syntactic argument on the *NP* category to map the dependency to, so the dependency cannot be created (and is in fact missing from CCGbank).

4.2 Lexical Rules and Zero Morphemes

The CCGbank PSG extension is closely related to the zero morpheme categories proposed by Aone and Wittenburg (1990), which they suggest be compiled into unary type-changing rules for processing. At first glance, it seems that conceptualising the rules as zero morphemes offers a way to locate them in the lexicon, avoiding the linguistic difficulties of having a language-specific grammar. However, CCG aims to provide a transparent interface between the surface form and the semantic analysis, so epsilon categories, traces, movement rules and other unrealised structures are explicitly banned (Steedman, 2000).

From a processing standpoint, if zero morpheme categories are not compiled into phrase-structure rules, then they will complicate the category assignment phase considerably, since we can no longer assume that exactly one category will be assigned per word. We are not aware of any proposal for how this difficulty might be overcome.

Carpenter (1992) provides a different suggestion for how sparse modifier categories can be accommodated. His solution is to use meta-rules that systematically expand the lexicon, much like the lexical rules used in HPSG (Flickinger, 1987), which exploit structural regularities to ensure that the lexicon is more complete.

The problem with this is that it does not actually make the category set less sparse, so the supertagger's task is just as difficult. The only advantage is that its dictionary will be more complete. This is important, but does not solve the underlying inefficiency in the grammar: CCG categories have an over-extended domain of locality, because they cannot represent form and function simultaneously. This is why some type-changing mechanism is required.

5 Lexically Specified Type-Changing

This section describes our mechanism for lexically specifying the PSG rules in CCGbank. Figure 3 shows an example of a reduced relative clause analysed using our extension, *hat categories*.

CCGbank deploys a solution that achieves *form transparency* at the expense of *type transparency*, by allowing type-changing rules that are not lexically specified. One way to recover the lost type transparency would be to demand that lexical categories specify what type changing rule (if any) the category can eventually undergo. For instance, imagine we have two type-changing rules we wish to include in our grammar:

a) $S[ng]\backslash NP \Rightarrow NP\backslash NP$

b) $S[ng]\backslash NP \Rightarrow VP\backslash VP$[2]

With these two rules, there will be three ways the $S[ng]\backslash NP$ category might behave in a derivation. What we need are two extra categories to control this:

1. $S[ng]\backslash NP$ only allows combinatory rules.
2. $(S[ng]\backslash NP)^a$ allows rule a, but not rule b.
3. $(S[ng]\backslash NP)^b$ allows rule b, but not rule a.

Instead of encoding a reference to a rule, we encode the production rule itself in the category.

[2] $S\backslash NP$ is occasionally abbreviated as *VP*.

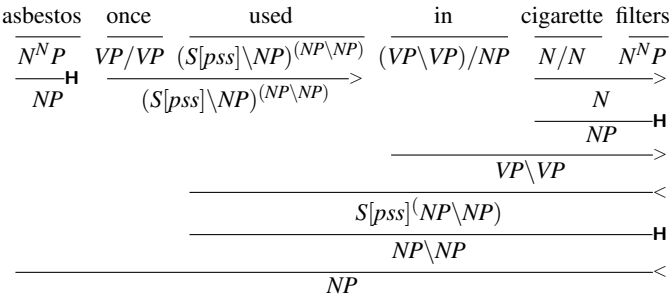


Figure 3: Analysis of a reduced relative clause with lexicalised type-changing.

$$\begin{bmatrix} \text{CAT} & (S[ng]\backslash NP)^{NP\backslash NP}/NP \\ \text{RES} & \begin{bmatrix} \text{CAT} & (S[ng]\backslash NP)^{NP\backslash NP} \\ \text{RES} & \begin{bmatrix} \text{CAT} & S \\ \text{FEAT} & ng \end{bmatrix} \\ \text{ARG} & NP_1 \\ \text{DIR} & \backslash \\ \text{HAT} & NP\backslash NP_1 \end{bmatrix} \\ \text{ARG} & NP \\ \text{DIR} & / \\ \text{HAT} & [\text{-}] \end{bmatrix}$$

Figure 4: AVM of a hat category.

This allows us to remove the rule from the grammar. Since the bottom of the production will always be the category itself, we can just specify how the category can be rewritten:

1. $S[ng]\backslash NP$ can be combined, but not rewritten.
2. $(S[ng]\backslash NP)^{NP\backslash NP}$ can be rewritten as $NP\backslash NP$.
3. $(S[ng]\backslash NP)^{VP\backslash VP}$ can be rewritten as $VP\backslash VP$.

We refer to the superscript category as a *hat* category, as a reference to the notation, but also to denote the fact that it allows the category to perform a different function, or put a different 'hat' on. Categories that have a hat specified are referred to as *hatted* categories.

5.1 Changes to Category Objects

Figure 4 shows an AVM representation of the $(S[ng]\backslash NP)^{NP\backslash NP}/NP$ category. A field, labelled *hat*, has been added to store the destination category of the result, $NP\backslash NP$. The NP argument in the hat category is co-indexed with the NP argument in the hatted category. The NP argument is also co indexed with the result of the destination category, reflecting the fact that the $NP\backslash NP$ category is a modifier, whose head will be the head of its argument.

Hat categories are handled the same as any other field during unification. If the two hat fields cannot be unified, unification fails; and if one hat field has an empty value, it inherits the value of the hat field of the other category when unification succeeds. CCG already requires a unification process for agreement features (Hockenmaier and Steedman, 2007); the hat categories we have introduced behave identically.

As Figure 4 shows, hat categories can be added to inner results, allowing arguments to be applied before the type-changing rule. We add a restriction that prevents categories with an outermost hat field from applying arguments — essentially equivalent to stipulating that the slash in a category like $S[ng]\backslash NP$ must have a null mode.

We also stipulate that only adjuncts may apply hatted arguments, which can also be lexically represented by assuming that all non-adjunct categories have a null value in their hat field, causing unification with a hatted category to fail.

Together, these restrictions ensure that the unary rule is used. The hatted category cannot function as a non-hatted category, because it cannot use its own arguments, and cannot be used as an argument of another category. This prevents hat categories from forming categories that are functionally *disjunctive*: the notation cannot be used to simulate something like an optional argument.

5.2 The Type-Change Rule

To replace the 204 PSG rules in CCGbank, we only need to introduce one extra schematic rule into the grammar:

$$X^Y \Rightarrow Y \tag{5}$$

This rule simply unpacks the category, performing the lexically specified type-change.

5.3 Generative Power

Because hat fields are only transmitted when categories are successfully unified, there is no way to produce a novel $X \Rightarrow Y$ unary production during a derivation. This means that any derivation that can be produced using the schematic type-change rule we have added to the grammar can be produced by adding a set of unary phrase-structure rules instead — ensuring that we do not require any extra generative power than is required to parse CCGbank.

The hat categories do increase the strong generative power of a CCG grammar that does not include the CCGbank type-changing rules. We suggest that this is desirable, in line with Joshi's (1999) argument that formalisms should be designed to have the maximum expressivity while maintaining the minimum weak generative power necessary to produce the constructions that have been observed in natural language.

6 Lexicalising Type-raising

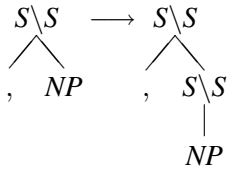
So far, we have focused on replacing the phrase-structure rules added to CCGbank, which are not part of the CCG linguistic theory. However, the theory does include some type-changing rules, referred to as *type-raising*. Forward and backward type-raising are used to transform a category X into the logically equivalent categories $T/(T\backslash X)$ and $T\backslash(T/X)$ respectively.

Type-raising is generally described as a lexical operation, rather than a grammatical rule, because only certain language specific combinations of T and X produce valid type-raise categories. However, no specific mechanism for controlling type-raising has been proposed.

Hat categories are an obvious candidate for this, so we perform an additional set of experiments which lexicalise the type-raising rules in CCGbank, in addition to the PSG rules.

7 Adapting CCGbank

This section describes how we converted CCGbank's PSG rules into analyses that used hat categories. Most of the PSG rules are unary, which meant that our changes were limited to adding hat categories to the child of the unary production and its subtree. The binary PSG rules that we converted effectively just used punctuation as a cue for a unary type-change, as seen in the extraposition rule in Figure 2. These were handled by adding an extra node for the punctuation application, leaving a unary production:



(6)

An alternative analysis would be to assign the punctuation mark a category to perform the type-change — in this case, $(S\backslash S)/NP$. However, this analysis will be unreliable for many genres, where punctuation is used inconsistently, so we preferred that hat category analysis, which we found produced slightly better results.

We used the same method to convert cases where CCGbank used conjunctions to cue a type-change, where the Penn Treebank conversion process produced a derivation where two sides of a coordination had different categories. There were 90 such conjunction coercion rules, which we have not counted amongst the 204 PSG rules, since they are ultimately caused by conversion noise.

The main complication when adapting CCGbank was the fact that CCG node labels are interdependent through a derivation. If one node label is changed, its immediate children have to change node label too, and the changes must be propagated further from there.

Since the dependency between the parent and its two children is different for each combinator, our node change rules determine the rule used for the original production, and then invoke the appropriate replacement rule. In general, the rules find the result (A_r) and argument (A_a) of the original parent A and replace them with the appropriate part of the new parent B. If one of the children is an adjunct category, a different rule is used. The node change rules for forward combinators are:

App	A/Y	Y	$\Rightarrow$	B/Y	Y
Comp	A_r/Y	Y/A_a	$\Rightarrow$	B_r/Y	Y/B_a
Adj. app	A/A	A	$\Rightarrow$	B/B	B
Adj. comp	A_r/A_r	A_r/A_a	$\Rightarrow$	B_r/B_r	B_r/B_a

The translation rules for backward and crossed combinators are directly analogous, with the slashes permuted appropriately.

8 Adapting the CCG Parser

We took the standard 1.02 release of the C&C parser Clark and Curran (2007) and implemented the changes required for lexically specified type-changing.

	Section 00							Section 23						
	LP	LR	LF	LF_{auto}	sent	cat	cov	LP	LR	LF	LF_{auto}	sent	cat	cov
CCGbank derivs	87.18	86.31	86.74	84.78	35.15	94.04	99.06	87.76	86.99	87.38	84.84	37.03	94.26	99.63
Hat derivs	86.64	86.91	86.77	84.44	35.03	93.27	99.53	86.94	87.26	87.10	84.76	36.62	93.35	99.71
Hat+TR derivs	86.58	86.87	86.73	84.16	34.47	93.10	99.63	86.83	87.16	87.00	84.67	36.73	93.17	99.75
CCGbank hybrid	88.07	86.49	87.27	85.30	35.94	94.16	99.06	88.36	87.02	87.68	85.27	36.74	94.33	99.63
Hat hybrid	87.30	86.94	87.12	84.85	35.40	93.31	99.53	87.26	87.03	87.15	84.79	36.25	93.24	99.71
Hat+TR hybrid	85.79	85.30	85.55	83.13	31.90	92.48	99.63	85.93	85.65	85.79	83.39	32.03	92.46	99.75

Table 1: Labelled Precision, Recall and F-measure, coverage results on Section 00 and Section 23.

The most significant change was building hat passing and unification into the existing unification engine. For many parsers, this would have been straightforward since they already support unification with complex feature structures. However, one of the advantages of CCGbank is that the unification required is quite simple, which is one of the reasons why the C&C parser is very fast. We would estimate that adding hat passing doubled the complexity of the unification engine.

The second step was to add support for hat passing to all of the existing combinators, because they do not use the unification engine to construct the result category. Since each of the combinators is hard-coded for speed, this was time-consuming and error prone. However, we created a detailed set of regression tests for the new versions which greatly reduced our development time.

Finally, we needed to turn off the existing unary rules in the parser, and add the simple additional type-change rule.

9 Setting Dictionary Thresholds

The main parameterisation we performed on the development section was to tune the K parameter of the parser, which controls the frequency at which a word's *tag dictionary* is used during supertagging. For words more frequent than K, the supertagger is restricted to choosing between categories that have been assigned to the word in the training data. Otherwise, the POS dictionary is used instead. The K parameter has multiple values, because the supertagger and parser are integrated such that the supertagger initially supplies only a narrow beam of categories to the parser, which is widened if parsing fails.

Since we have made the category set larger, the default values of $K = 20, 20, 20, 20, 150$ produces poor performance, up to 1.5% lower than the figures we report in Table 1. We set the K parameter to $50, 300, 80, 80, 3000$. We investigated the effect of this setting on the original model, and found that it had little effect, so we continued using the default values for the original model.

We also experimented with altering the β values for the hat parser, which did not improve the performance of the state-of-the-art parsing models.

	Section 00		Section 23	
Training	Gold	Auto	Gold	Auto
CCGbank derivs	399	413	639	544
Hat derivs	552	566	1070	827
Hat+TR derivs	718	677	1072	906
CCGbank hybrid	369	379	564	480
Hat hybrid	505	513	921	678
Hat+TR hybrid	645	601	913	785

Table 2: Parse speeds in words per second.

	Original		Hat	
	Types	Frequency	Types	Frequency
Binary CCG	2,714	1,097,809	3,840	1,097,358
Type-raise	52	3,998	52	3,996
Unhat	0	0	241	161,069
Binary PSG	215	1,615	74	172
Unary PSG	157	159,663	0	0

Table 3: Production types and frequencies.

10 Parsing Results

The left side of Table 1 shows our performance on the development data, Section 00. All of the dependency results we report refer to the original dependencies distributed with CCGbank. To ensure our results were comparable, we produced a mapping table of dependency labels from sections 02-21, used for parser training. The table maps the dependency labels in our corpus to the most frequent label assigned to matching dependencies in CCGbank. The correct label is assigned 99.94% of the time. The hat categories move to the the lexicon information that used to be represented in the grammar, resulting in a larger, more informative

category set, making the category accuracies (the *cat* column in the table) not comparable.

We experimented with two of the parsing models described by Clark and Curran (2007). The *derivs* model uses features calculated over the derivations, while the *hybrid* model uses features calculated on the dependency structures. However, unlike the *deps* model Clark and Curran (2007) describe, the *hybrid* model uses two sets of derivation-based constraints. One set are the normal form constraints, as described by Eisner (1996). It also uses constraints that prevent the parser from using productions that were not seen in the training data. The hybrid model is slightly more accurate, but also slightly slower, because the dependency-based decoding is less efficient.

All of the systems were within 0.5% in accuracy on the development set, with one exception. The HAT+TR version performed very poorly with the hybrid model, while its performance with the derivs model was comparable to the other systems. The same drop in performance occurs on the evaluation set. We do not currently have a convincing explanation for this, but we presume it is the result of some unforeseen interaction between the removal of the type-raising rules from the grammar and the dependency-based features.

The accuracy results on the test data, Section 23, saw similar trends, except that the gap between the hat systems and the original CCGbank increased slightly. The CCGbank hybrid model was only 0.1% more accurate than the HAT hybrid model on Section 00, but is 0.5% more accurate on Section 23.

Table 2 compares the parse speeds for the lexicalised hat corpora against a parser trained on the original CCGbank, using the two models. Exactly the same settings were used to obtain parse times as were used in the accuracy experiments. The experiments were all performed on a single core 2.6GHz Pentium 4 Xeon. Speeds are reported as words parsed per second.

On both Section 00 and Section 23, with both the derivs and hybrid models, the HAT system was substantially faster than the original parser. The HAT+TR system was faster than the HAT system using automatic POS tags, and slightly faster on Section 00.

The hat categories allow quite favourable tradeoffs between speed and accuracy to be made. The original models allow us to parse with automatic POS tags at 480 words per second with 85.27% accuracy with the hybrid model, or at 544 words per second with 84.86% accuracy using the derivs model. Using the HAT derivs model, we could instead parse at 827 words per second with 84.76% accuracy, or at 906 words per second and 84.67% accuracy using the HAT+TR system.

In summary, the HAT and CCGbank derivs models are equivalent in accuracy, but the HAT version is 52% faster. The CCGbank hybrid model remains the most accurate, but there will also be many tasks where the 88% improvement in speed will make it worth using the HAT+TR derivs parser instead of the CCGbank hybrid model, at a cost of 0.6% accuracy.

11 Corpus Statistics

Table 3 shows the number of types and the number of occurrences of CCG combinatory rules and PSG rules occurred in CCGbank and the hat corpus.

The hat corpus removes almost all unlicensed productions, leaving only a long tail of rare productions that are the result of noisy derivations. These productions are generally symptomatic of problematic analyses, and are difficult to address automatically because they do not conform to any consistent pattern. We have omitted the hat+TR corpus in these figures, because it only differs from the the hat corpus with respect to type-raising productions.

Lexicalising the corpus increases the number of categories required substantially. There are 407 categories that occur 10 or more times in the training section of CCGbank. The equivalent figure for the HAT corpus is 507, and for the HAT+TR corpus it is 540.

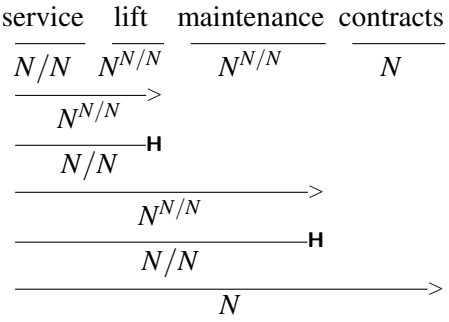
12 Cleaner Analyses with Hat Categories

The lexicalised type-changing scheme we have proposed offers many opportunities for favourable analyses, because it allows form and function to be represented simultaneously. However, we have limited our changes to replacing the existing CCGbank non-combinatory rules. This allows us to compare the two strategies for controlling modifier category proliferation more closely, but still offers some improved analyses.

The most frequent unary production in CCGbank, the $N \Rightarrow NP$ rule, ensures that nominals can always take the N category, so adjectives seldom need to be assigned NP/NP. Because ad-

jectives and nouns are open class, and bare noun phrases are fairly common, this reduction in category sparseness is quite important.

Lexicalising the type changing rule forces the head noun to acquire a different category, but does ensure that its modifiers can attach at the *N* level — which is also more linguistically desirable:



This analysis also prevents the extreme category proliferation problem caused by left-branching noun phrases:

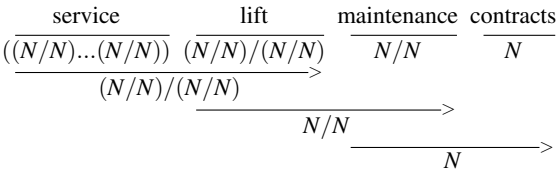


Figure 3 shows a more typical example of an improved analysis. The non-finite clause is functioning as an adnominal, but its modifier is able to select its canonical category.

One of the advantages of the CCGbank phrase-structure rules is that they allow the corpus to include derivations for which no valid CCG parse can be formed. The C&C parser has difficulty taking advantage of these extra sentences, however, because only so many of the arbitrary binary PSG rules can be added to the grammar without making it too ambiguous. Once these rules are lexicalised, the categories that produce them can be added to the lexicon as unexceptional, albeit rare, cases.

13 Conclusion

Lexicalised grammars represent most of the information in a derivation with a sequence of lexical categories. Traditional CCG analyses require redundancy between categories whenever there is nested modification, which suggests that such analyses will encounter sparse data problems.

While the addition of phrase-structure rules prevents this proliferation of modifier categories, it does so at a high price. The bulk of the type-changing rules in CCGbank are not implemented in the C&C parser, because to do so would increase the ambiguity in the grammar enormously.

CCG parsers must carefully manage ambiguity, because there are many ways to bracket the same CCG derivation. Even with a restricted set of PSG rules, the C&C parser experiences very large chart sizes. In addition to making the grammar more ambiguous, the PSG rules make it less theoretically sound, and more difficult to produce semantic analyses from the parser's output.

We have show how CCG analyses can be fully lexicalised in a way that closely mirrors the introduction of phrase-structure rules. The result is a corpus that produces faster, accurate parsers, is well suited for domain adaptation, and allows for more transparent semantic analysis. We can also use the same mechanism to lexically specify type-raising, the first concrete proposal to handle type-raising as a lexical transformation we are aware of.

From an immediate, empirical perspective, we have substantially improved the parsing speed of what is already the fastest deep parser available. Improvements in parsing efficiency are important in making parsing a practical technology, since the volume of text we have available for processing is growing even faster than the processing resources we have available.

Acknowledgements

We would like to thank Stephen Clark and the anonymous reviewers for EMNLP and the Grammar Engineering Across Frameworks workshop for their valuable feedback. This work was supported by the Australian Research Council under Discovery Project DP0665973.

References

Chinatsu Aone and Kent Wittenburg. 1990. Zero morphemes in unification-based combinatory categorial grammar. In *ACL*, pages 188–193.

Jason Baldridge and Geert-Jan Kruijff. 2003. Multi-Modal Combinatory Categorial Grammar. In *Proceedings of the European Association of Computational Linguistics (EACL)*.

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

Yehoshua Bar-Hillel. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

Bob Carpenter. 1992. *Categorial grammars, lexical rules, and the English predicative*, chapter 3. Oxford University Press.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Jason Eisner. 1996. Efficient normal-form parsing for Combinatory Categorial Grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL-96)*, pages 79–86. Santa Cruz, CA, USA.

Dan Flickinger. 1987. *Lexical Rules in the Hierarchical Lexicon*. Ph.D. thesis, Stanford University, Stanford, CA.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Third LREC*, pages 1974–1981.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Aravind K. Joshi. 1999. Explorations of a domain of locality: Lexicalized tree-adjoining grammar. In *CLIN*.

Mitchell Marcus, Beatrice Santorini, and Mary Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Michael Moortgat. 1997. Categorial type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 2, pages 93–177. Elsevier, Amsterdam and MIT Press, Cambridge MA.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

Mark Steedman and Jason Baldridge. 2007. Combinatory categorial grammar. In Robert Borsley and Kersti Borjars, editors, *Non-Transformational Syntax*. Blackwells.

Bilingually-Constrained (Monolingual) Shift-Reduce Parsing

Liang Huang
Google Research
1350 Charleston Rd.
Mountain View, CA 94043, USA
lianghuang@google.com
liang.huang.sh@gmail.com

Wenbin Jiang and **Qun Liu**
Key Lab. of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
jiangwenbin@ict.ac.cn

Abstract

Jointly parsing two languages has been shown to improve accuracies on either or both sides. However, its search space is much bigger than the monolingual case, forcing existing approaches to employ complicated modeling and crude approximations. Here we propose a much simpler alternative, *bilingually-constrained monolingual parsing*, where a source-language parser learns to exploit reorderings as additional observation, but *not* bothering to build the target-side tree as well. We show specifically how to enhance a shift-reduce dependency parser with alignment features to resolve shift-reduce conflicts. Experiments on the bilingual portion of Chinese Treebank show that, with just 3 bilingual features, we can improve parsing accuracies by 0.6% (absolute) for both English and Chinese over a state-of-the-art baseline, with negligible (∼6%) efficiency overhead, thus much faster than biparsing.

1 Introduction

Ambiguity resolution is a central task in Natural Language Processing. Interestingly, not all languages are ambiguous in the same way. For example, prepositional phrase (PP) attachment is (notoriously) ambiguous in English (and related European languages), but is strictly unambiguous in Chinese and largely unambiguous Japanese; see

(1a) I [saw Bill] [with a telescope].
wo [*yong wangyuanjin*] [*kandao le Bi'er*].
"I used a telescope to see Bill."

(1b) I saw [Bill [with a telescope]].
wo kandao le [[*na wangyuanjin*] *de Bi'er*].
"I saw Bill who had a telescope at hand."

Figure 1: PP-attachment is unambiguous in Chinese, which can help English parsing.

Figure 1 for an example.[1] It is thus intuitive to use two languages for better disambiguation, which has been applied not only to this PP-attachment problem (Fossum and Knight, 2008; Schwartz et al., 2003), but also to the more fundamental problem of syntactic parsing which subsumes the former as a subproblem. For example, Smith and Smith (2004) and Burkett and Klein (2008) show that joint parsing (or reranking) on a bitext improves accuracies on either or both sides by leveraging bilingual constraints, which is very promising for syntax-based machine translation which requires (good-quality) parse trees for rule extraction (Galley et al., 2004; Mi and Huang, 2008).

However, the search space of joint parsing is inevitably much bigger than the monolingual case,

[1] Chinese uses word-order to disambiguate the attachment (see below). By contrast, Japanese resorts to case-markers and the unambiguity is limited: it works for the "V or N" attachment ambiguities like in Figure 1 (see (Schwartz et al., 2003)) but not for the "N_1 or N_2" case (Mitch Marcus, p.c.).

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1222–1231,
Singapore, 6-7 August 2009.

forcing existing approaches to employ complicated modeling and crude approximations. Joint parsing with a simplest synchronous context-free grammar (Wu, 1997) is $O(n^6)$ as opposed to the monolingual $O(n^3)$ time. To make things worse, languages are *non-isomorphic*, i.e., there is no 1-to-1 mapping between tree nodes, thus in practice one has to use more expressive formalisms such as synchronous tree-substitution grammars (Eisner, 2003; Galley et al., 2004). In fact, rather than joint parsing per se, Burkett and Klein (2008) resort to separate monolingual parsing and *bilingual reranking* over k^2 tree pairs, which covers a tiny fraction of the whole space (Huang, 2008).

We instead propose a much simpler alternative, *bilingually-constrained monolingual parsing*, where a source-language parser is extended to exploit the reorderings between languages as additional observation, but *not* bothering to build a tree for the target side simultaneously. To illustrate the idea, suppose we are parsing the sentence

(1) I saw Bill [$_{\text{PP}}$ with a telescope].

which has 2 parses based on the attachment of PP:

(1a) I [saw Bill] [$_{\text{PP}}$ with a telescope].

(1b) I saw [Bill [$_{\text{PP}}$ with a telescope]].

Both are possible, but with a Chinese translation the choice becomes clear (see Figure 1), because a Chinese PP always immediately precedes the phrase it is modifying, thus making PP-attachment strictly unambiguous.[2] We can thus use Chinese to help parse English, i.e., whenever we have a PP-attachment ambiguity, we will consult the Chinese translation (from a bitext), and based on the alignment information, decide where to attach the English PP. On the other hand, English can help Chinese parsing as well, for example in deciding the scope of relative clauses which is unambiguous in English but ambiguous in Chinese.

This method is much simpler than joint parsing because it remains *monolingual* in the backbone, with alignment information merely as soft evidence, rather than hard constraints since automatic word alignment is far from perfect. It is thus straightforward to implement within a monolingual parsing algorithm. In this work we choose shift-reduce dependency parsing for its simplicity and efficiency. Specifically, we make the following contributions:

- we develop a baseline shift-reduce dependency parser using the less popular, but classical, "arc-standard" style (Section 2), and achieve similar state-of-the-art performance with the the dominant but complicated "arc-eager" style of Nivre and Scholz (2004);
- we propose bilingual features based on word-alignment information to prefer "target-side contiguity" in resolving shift-reduce conflicts (Section 3);
- we verify empirically that shift-reduce conflicts are the major source of errors, and correct shift-reduce decisions strongly correlate with the above bilingual contiguity conditions *even* with automatic alignments (Section 5.3);
- finally, with just three bilingual features, we improve dependency parsing accuracy by 0.6% for both English and Chinese over the state-of-the-art baseline with negligible ($\sim$6%) efficiency overhead (Section 5.4).

2 Simpler Shift-Reduce Dependency Parsing with Three Actions

The basic idea of classical shift-reduce parsing from compiler theory (Aho and Ullman, 1972) is to perform a left-to-right scan of the input sentence, and at each step, choose one of the two actions: either *shift* the current word onto the stack, or *reduce* the top two (or more) items on the stack, replacing them with their combination. This idea has been applied to constituency parsing, for example in Sagae and Lavie (2006), and we describe below a simple variant for dependency parsing similar to Yamada and Matsumoto (2003) and the "arc-standard" version of Nivre (2004).

2.1 The Three Actions

Basically, we just need to split the reduce action into two symmetric (sub-)actions, reduce$_{\text{L}}$ and reduce$_{\text{R}}$, depending on which one of the two

[2] to be precise, in Fig. 1(b), the English PP is translated into a Chinese relative clause, but nevertheless all phrasal modifiers attach to the immediate right in Mandarin Chinese.

	stack	queue	arcs
previous	S	$w_i \vert Q$	A
shift	$S \vert w_i$	Q	A
previous	$S \vert s_{t-1} \vert s_t$	Q	A
reduce_L	$S \vert s_t$	Q	$A \cup \{(s_t, s_{t-1})\}$
reduce_R	$S \vert s_{t-1}$	Q	$A \cup \{(s_{t-1}, s_t)\}$

Table 1: Formal description of the three actions. Note that shift requires non-empty queue while reduce requires at least two elements on the stack.

items becomes the head after reduction. More formally, we describe a parser configuration by a tuple $\langle S, Q, A \rangle$ where S is the stack, Q is the queue of remaining words of the input, and A is the set of dependency arcs accumulated so far.[3] At each step, we can choose one of the three actions:

1. shift: move the head of (a non-empty) queue Q onto stack S;
2. reduce_L: combine the top two items on the stack, s_t and s_{t-1} ($t \geq 2$), and replace them with s_t (as the head), and add a left arc (s_t, s_{t-1}) to A;
3. reduce_R: combine the top two items on the stack, s_t and s_{t-1} ($t \geq 2$), and replace them with s_{t-1} (as the head), and add a right arc (s_{t-1}, s_t) to A.

These actions are summarized in Table 1. The initial configuration is always $\langle \emptyset, w_1 \ldots w_n, \emptyset \rangle$ with empty stack and no arcs, and the final configuration is $\langle w_j, \emptyset, A \rangle$ where w_j is recognized as the root of the whole sentence, and A encodes a spanning tree rooted at w_j. For a sentence of n words, there are exactly $2n - 1$ actions: n shifts and $n - 1$ reductions, since every word must be pushed onto stack once, and every word except the root will eventually be popped in a reduction. The time complexity, as other shift-reduce instances, is clearly $O(n)$.

2.2 Example of Shift-Reduce Conflict

Figure 2 shows the trace of this paradigm on the example sentence. For the first two configurations

[3] a "configuration" is sometimes called a "state" (Zhang and Clark, 2008), but that term is confusing with the states in shift-reduce LR/LL parsing, which are quite different.

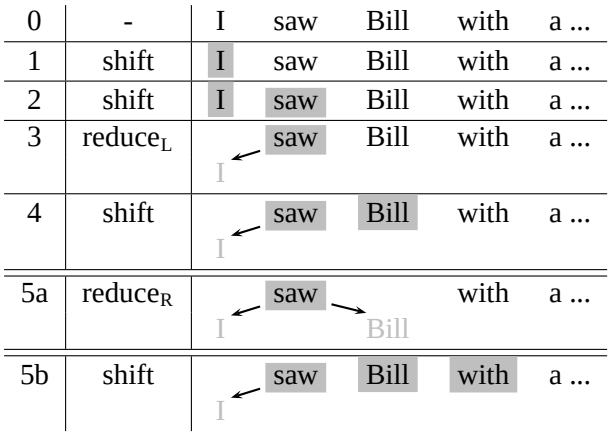


Figure 2: A trace of 3-action shift-reduce on the example sentence. Shaded words are on stack, while gray words have been popped from stack. After step (4), the process can take either (5a) or (5b), which correspond to the two attachments (1a) and (1b) in Figure 1, respectively.

(0) and (1), only shift is possible since there are not enough items on the stack for reduction. At step (3), we perform a reduce_L, making word "I" a modifier of "saw"; after that the stack contains a single word and we have to shift the next word "Bill" (step 4). Now we face a *shift-reduce conflict*: we can either combine "saw" and "Bill" in a reduce_R action (5a), or shift "Bill" (5b). We will use features extracted from the configuration to resolve the conflict. For example, one such feature could be a bigram $s_t \circ s_{t-1}$, capturing how likely these two words are combined; see Table 2 for the complete list of feature templates we use in this baseline parser.

We argue that this kind of shift-reduce conflicts are the major source of parsing errors, since the other type of conflict, reduce-reduce conflict (i.e., whether left or right) is relatively easier to resolve given the part-of-speech information. For example, between a noun and an adjective, the former is much more likely to be the head (and so is a verb vs. a preposition or an adverb). Shift-reduce resolution, however, is more non-local, and often involves a triple, for example, (saw, Bill, with) for a typical PP-attachment. On the other hand, if we indeed make a wrong decision, a reduce-reduce mistake just flips the head and the modifier, and often has a more local effect on the shape of the tree, whereas a shift-reduce mistake always leads

Type	Features		
Unigram	s_t	$T(s_t)$	$s_t \circ T(s_t)$
	s_{t-1}	$T(s_{t-1})$	$s_{t-1} \circ T(s_{t-1})$
	w_i	$T(w_i)$	$w_i \circ T(w_i)$
Bigram	$s_t \circ s_{t-1}$	$T(s_t) \circ T(s_{t-1})$	$T(s_t) \circ T(w_i)$
	$T(s_t) \circ s_{t-1} \circ T(s_{t-1})$	$s_t \circ s_{t-1} \circ T(s_{t-1})$	$s_t \circ T(s_t) \circ T(s_{t-1})$
	$s_t \circ T(s_t) \circ s_{t-1}$	$s_t \circ T(s_t) \circ s_{t-1} \circ T(s_{t-1})$	
Trigram	$T(s_t) \circ T(w_i) \circ T(w_{i+1})$	$T(s_{t-1}) \circ T(s_t) \circ T(w_i)$	$T(s_{t-2}) \circ T(s_{t-1}) \circ T(s_t)$
	$s_t \circ T(w_i) \circ T(w_{i+1})$	$T(s_{t-1}) \circ s_t \circ T(w_i)$	
Modifier	$T(s_{t-1}) \circ T(lc(s_{t-1})) \circ T(s_t)$	$T(s_{t-1}) \circ T(rc(s_{t-1})) \circ T(s_t)$	$T(s_{t-1}) \circ T(s_t) \circ T(lc(s_t))$
	$T(s_{t-1}) \circ T(s_t) \circ T(rc(s_t))$	$T(s_{t-1}) \circ T(lc(s_{t-1})) \circ s_t$	$T(s_{t-1}) \circ T(rc(s_{t-1})) \circ s_t$
	$T(s_{t-1}) \circ s_t \circ T(lc(s_t))$		

Table 2: Feature templates of the baseline parser. s_t, s_{t-1} denote the top and next to top words on the stack; w_i and w_{i+1} denote the current and next words on the queue. $T(\cdot)$ denotes the POS tag of a given word, and $lc(\cdot)$ and $rc(\cdot)$ represent the leftmost and rightmost child. Symbol $\circ$ denotes feature conjunction. Each of these templates is further conjoined with the 3 actions shift, reduce$_{\text{L}}$, and reduce$_{\text{R}}$.

to vastly incompatible tree shapes with crossing brackets (for example, [saw Bill] vs. [Bill with a telescope]). We will see in Section 5.3 that this is indeed the case in practice, thus suggesting us to focus on shift-reduce resolution, which we will return to with the help of bilingual constraints in Section 3.

2.3 Comparison with Arc-Eager

The three action system was originally described by Yamada and Matsumoto (2003) (although their methods require multiple passes over the input), and then appeared as "arc-standard" in Nivre (2004), but was argued against in comparison to the four-action "arc-eager" variant. Most subsequent works on shift-reduce or "transition-based" dependency parsing followed "arc-eager" (Nivre and Scholz, 2004; Zhang and Clark, 2008), which now becomes the dominant style. But we argue that "arc-standard" is preferable because:

1. in the three action "arc-standard" system, the stack always contains a list of *unrelated* subtrees recognized so far, with no arcs between any of them, e.g. (I← saw) and (Bill) in step 4 of Figure 2), whereas the four action "arc-eager" style can have left or right arrows between items on the stack;

2. the semantics of the three actions are atomic and disjoint, whereas the semantics of 4 actions are *not* completely disjoint. For example, their Left action assumes an implicit Reduce of the left item, and their Right action assumes an implicit Shift. Furthermore, these two actions have non-trivial preconditions which also causes the next problem (see below). We argue that this is rather complicated to implement.

3. the "arc-standard" scan always succeeds, since at the end we can always reduce with empty queue, whereas the "arc-eager" style sometimes goes into deadends where no action can perform (prevented by preconditions, otherwise the result will not be a well-formed tree). This becomes parsing failures in practice (Nivre and Scholz, 2004), leaving more than one fragments on stack.

As we will see in Section 5.1, this simpler arc-standard system performs equally well with a state-of-the-art arc-eager system (Zhang and Clark, 2008) on standard English Treebank parsing (which is never shown before). We argue that all things being equal, this simpler paradigm should be preferred in practice. [4]

2.4 Beam Search Extension

We also enhance deterministic shift-reduce parsing with beam search, similar to Zhang and Clark (2008), where k configurations develop in parallel. Pseudocode 1 illustrates the algorithm, where we keep an agenda $\mathbf{V}$ of the current active configurations, and at each step try to extend them by applying one of the three actions. We then dump the best k new configurations from the buffer back

[4] On the other hand, there are also arguments for "arc-eager", e.g., "incrementality"; see (Nivre, 2004; Nivre, 2008).

Pseudocode 1 beam-search shift-reduce parsing.

Input: POS-tagged word sequence $w_1 \dots w_n$
$start \leftarrow \langle \emptyset, w_1 \dots w_n, \emptyset \rangle$ ▷ initial config: empty stack, no arcs
$\mathbf{V} \leftarrow \{start\}$ ▷ initial agenda
for $step \leftarrow 1 \dots 2n-1$ **do**
  $\mathbf{BUF} \leftarrow \emptyset$ ▷ buffer for new configs
  for each $config$ in agenda $\mathbf{V}$ **do**
    for $act \in \{\text{shift}, \text{reduce}_\text{L}, \text{reduce}_\text{R}\}$ **do**
      if act is applicable to $config$ **then**
        $next \leftarrow$ apply act to $config$
        insert $next$ into buffer $\mathbf{BUF}$
  $\mathbf{V} \leftarrow$ top k configurations of $\mathbf{BUF}$
Output: the tree of the best config in $\mathbf{V}$

into the agenda for the next step. The complexity of this algorithm is $O(nk)$, which subsumes the determinstic mode as a special case ($k = 1$).

2.5 Online Training

To train the parser we need an "oracle" or gold-standard action sequence for gold-standard dependency trees. This oracle turns out to be *non-unique* for the three-action system (also non-unique for the four-action system), because left dependents of a head can be reduced either before or after all right dependents are reduced. For example, in Figure 2, "I" is a left dependent of "saw", and can in principle wait until "Bill" and "with" are reduced, and then finally combine with "saw". We choose to use the heuristic of "shortest stack" that always prefers reduce_L over shift, which has the effect that all left dependents are first recognized inside-out, followed by all right dependents, also inside-out, which coincides with the head-driven constituency parsing model of Collins (1999).

We use the popular online learning algorithm of structured perceptron with parameter averaging (Collins, 2002). Following Collins and Roark (2004) we also use the "early-update" strategy, where an update happens whenever the gold-standard action-sequence falls off the beam, with the rest of the sequence neglected. As a special case, for the deterministic mode, updates always co-occur with the first mistake made. The intuition behind this strategy is that future mistakes are often caused by previous ones, so with the parser on the wrong track, future actions become irrelevant for learning. See Section 5.3 for more discussions.

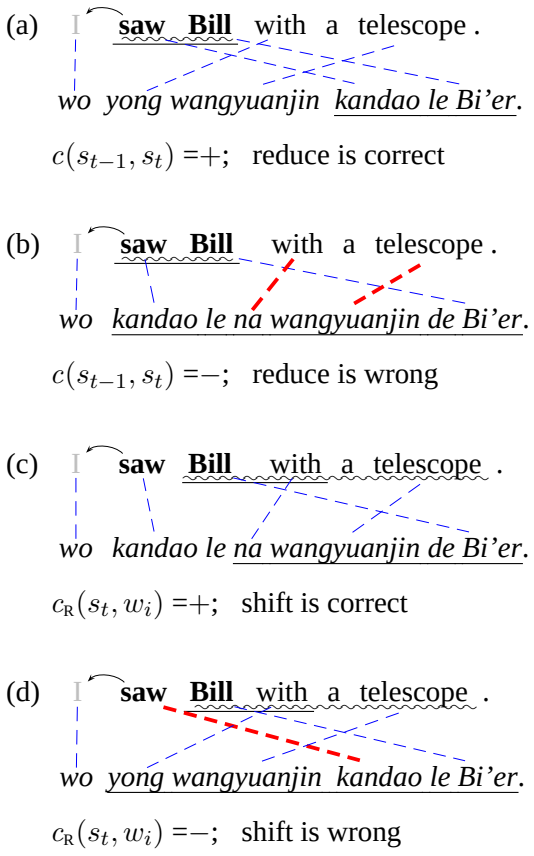


Figure 3: Bilingual contiguity features $c(s_{t-1}, s_t)$ and $c_\text{R}(s_t, w_i)$ at step (4) in Fig. 2 (facing a shift-reduce decision). Bold words are currently on stack while gray ones have been popped. Here the stack tops are s_t = **Bill**, s_{t-1} = **saw**, and the queue head is w_i = with; underlined texts mark the source and target spans being considered, and wavy underlines mark the *allowed spans* (Tab. 3). Red bold alignment links violate contiguity constraints.

3 Soft Bilingual Constraints as Features

As suggested in Section 2.2, shift-reduce conflicts are the central problem we need to address here. Our intuition is, whenever we face a decision whether to combine the stack tops s_{t-1} and s_t or to shift the current word w_i, we will consult the other language, where the word-alignment information would hopefully provide a preference, as in the running example of PP-attachment (see Figure 1). We now develop this idea into *bilingual contiguity features*.

3.1 A Pro-Reduce Feature $c(s_{t-1}, s_t)$

Informally, if the correct decision is a reduction, then it is likely that the corresponding words of s_{t-1} and s_t on the target-side should also form a contiguous span. For example, in Figure 3(a), the source span of a reduction is [saw .. Bill], which maps onto [*kandao . . . Bi'er*] on the Chinese side. This target span is contiguous, because no word within this span is aligned to a source word outside of the source span. In this case we say feature $c(s_{t-1}, s_t)$ =+, which encourages "reduce".

However, in Figure 3(b), the source span is still [saw .. Bill], but this time maps onto a much longer span on the Chinese side. This target span is discontiguous, since the Chinese words *na* and *wangyuanjin* are alinged to English "with" and "telescope", both of which fall outside of the source span. In this case we say feature $c(s_{t-1}, s_t)$ =−, which discourages "reduce" .

3.2 A Pro-Shift Feature $c_{\text{R}}(s_t, w_i)$

Similarly, we can develop another feature $c_{\text{R}}(s_t, w_i)$ for the shift action. In Figure 3(c), when considering shifting "with", the source span becomes [Bill .. with] which maps to [*na .. Bi'er*] on the Chinese side. This target span looks like discontiguous in the above definition with *wangyuanjin* aligned to "telescope", but we tolerate this case for the following reasons. There is a crucial difference between shift and reduce: in a shift, we do not know yet the subtree spans (unlike in a reduce we are always combining two well-formed subtrees). The only thing we are sure of in a shift action is that s_t and w_i will be combined *before* s_{t-1} and s_t are combined (Aho and Ullman, 1972), so we can tolerate any target word aligned to source word still in the queue, but do not allow any target word aligned to an already recognized source word. This explains the notational difference between $c_{\text{R}}(s_t, w_i)$ and $c(s_{t-1}, s_t)$, where subscript "R" means "right contiguity".

As a final example, in Figure 3(d), Chinese word *kandao* aligns to "saw", which is already recognized, and this violates the right contiguity. So $c_{\text{R}}(s_t, w_i)$ =−, suggesting that shift is probably wrong. To be more precise, Table 3 shows the formal definitions of the two features. We basically map a source span sp to its target span $M(sp)$, and check whether its reverse image back onto the source language $M^{-1}(M(sp))$ falls inside the allowed span ap. For $c_{\text{R}}(s_t, w_i)$, the allowed span extends to the right end of the sentence.[5]

feature f	source span sp	target span tp	allowed span ap
$c(s_{t-1}, s_t)$	$[s_{t-1}..s_t]$	$M(sp)$	$[s_{t-1}..s_t]$
$c_{\text{R}}(s_t, w_i)$	$[s_t..w_i]$	$M(sp)$	$[s_t..w_n]$
$f = +$	iff. $M^{-1}(M(sp)) \subseteq ap$		

Table 3: Formal definition of bilingual features. $M(\cdot)$ is maps a source span to the target language, and $M^{-1}(\cdot)$ is the reverse operation mapping back to the source language.

3.3 Variations and Implementation

To conclude so far, we have got two alignment-based features, $c(s_{t-1}, s_t)$ correlating with reduce, and $c_{\text{R}}(s_t, w_i)$ correlating with shift. In fact, the conjunction of these two features,

$$c(s_{t-1}, s_t) \circ c_{\text{R}}(s_t, w_i)$$

is another feature with even stronger discrimination power. If

$$c(s_{t-1}, s_t) \circ c_{\text{R}}(s_t, w_i) = + \circ -$$

it is strongly recommending reduce, while

$$c(s_{t-1}, s_t) \circ c_{\text{R}}(s_t, w_i) = - \circ +$$

is a very strong signal for shift. So in total we got three bilingual feature (templates), which in practice amounts to 24 instances (after cross-product with $\{-, +\}$ and the three actions). We show in Section 5.3 that these features do correlate with the correct shift/reduce actions in practice.

The naive implemention of bilingual feature computation would be of $O(kn^2)$ complexity in the worse case because when combining the largest spans one has to scan over the whole sentence. We envision the use of a clever datastructure would reduce the complexity, but leave this to future work, as the experiments (Table 8) show that

[5] Our definition implies that we only consider *faithful* spans to be contiguous (Galley et al., 2004). Also note that source spans include all dependents of s_t and s_{t-1}.

the parser is only marginally (∼6%) slower with the new bilingual features. This is because the extra work, with just 3 bilingual features, is not the bottleneck in practice, since the extraction of the vast amount of other features in Table 2 dominates the computation.

4 Related Work in Grammar Induction

Besides those cited in Section 1, there are some other related work on using bilingual constraints for grammar induction (rather than parsing). For example, Hwa et al. (2005) use simple heuristics to project English trees to Spanish and Chinese, but get discouraging accuracy results learned from those projected trees. Following this idea, Ganchev et al. (2009) and Smith and Eisner (2009) use constrained EM and parser adaptation techniques, respectively, to perform more principled projection, and both achieve encouraging results.

Our work, by constrast, never uses bilingual tree pairs not tree projections, and only uses word alignment alone to enhance a monolingual grammar, which learns to prefer target-side contiguity.

5 Experiments

5.1 Baseline Parser

We implement our baseline monolingual parser (in C++) based on the shift-reduce algorithm in Section 2, with feature templates from Table 2. We evaluate its performance on the standard Penn English Treebank (PTB) dependency parsing task, i.e., train on sections 02-21 and test on section 23 with automatically assigned POS tags (at 97.2% accuracy) using a tagger similar to Collins (2002), and using the headrules of Yamada and Matsumoto (2003) for conversion into dependency trees. We use section 22 as dev set to determine the optimal number of iterations in perceptron training. Table 4 compares our baseline against the state-of-the-art graph-based (McDonald et al., 2005) and transition-based (Zhang and Clark, 2008) approaches, and confirms that our system performs at the same level with those state-of-the-art, and runs extremely fast in the deterministic mode (k=1), and still quite fast in the beam-search mode (k=16).

parser	accuracy	secs/sent
McDonald et al. (2005)	90.7	0.150
Zhang and Clark (2008)	91.4	0.195
our baseline at k=1	90.2	0.009
our baseline at k=16	91.3	0.125

Table 4: Baseline parser performance on standard Penn English Treebank dependency parsing task. The speed numbers are not exactly comparable since they are reported on different machines.

	Training	Dev	Test
CTB Articles	1-270	301-325	271-300
Bilingual Paris	2745	273	290

Table 5: Training, dev, and test sets from bilingual Chinese Treebank à la Burkett and Klein (2008).

5.2 Bilingual Data

The bilingual data we use is the translated portion of the Penn Chinese Treebank (CTB) (Xue et al., 2002), corresponding to articles 1-325 of PTB, which have English translations with gold-standard parse trees (Bies et al., 2007). Table 5 shows the split of this data into training, development, and test subsets according to Burkett and Klein (2008). Note that not all sentence pairs could be included, since many of them are not one-to-one aligned at the sentence level. Our word-alignments are generated from the HMM aligner of Liang et al. (2006) trained on approximately 1.7M sentence pairs (provided to us by David Burkett, p.c.). This aligner outputs "soft alignments", i.e., posterior probabilities for each source-target word pair. We use a pruning threshold of 0.535 to remove low-confidence alignment links,[6] and use the remaining links as hard alignments; we leave the use of alignment probabilities to future work.

For simplicity reasons, in the following experiments we always supply gold-standard POS tags as part of the input to the parser.

5.3 Testing our Hypotheses

Before evaluating our bilingual approach, we need to verify empirically the two assumptions we made about the parser in Sections 2 and 3:

[6] and also removing notoriously bad links in $\{\text{the, a, an}\} \times \{de, le\}$ following Fossum and Knight (2008).

	sh ▷ re	re ▷ sh	sh-re	re-re
#	92	98	190	7
%	46.7%	49.7%	**96.4%**	**3.6%**

Table 6: [Hypothesis 1] Error distribution in the baseline model ($k = 1$) on English dev set. "sh ▷ re" means "should shift, but reduced". Shift-reduce conflicts overwhelmingly dominate.

1. (monolingual) shift-reduce conflict is the major source of errors while reduce-reduce conflict is a minor issue;

2. (bilingual) the gold-standard decisions of shift or reduce should correlate with contiguities of $c(s_{t-1}, s_t)$, and of $c_{\text{R}}(s_t, w_i)$.

Hypothesis 1 is verified in Table 6, where we count all the *first mistakes* the baseline parser makes (in the deterministic mode) on the English dev set (273 sentences). In shift-reduce parsing, further mistakes are often caused by previous ones, so only the first mistake in each sentence (if there is one) is easily identifiable;[7] this is also the argument for "early update" in applying perceptron learning to these incremental parsing algorithms (Collins and Roark, 2004) (see also Section 2). Among the 197 first mistakes (other 76 sentences have perfect output), the vast majority, 190 of them (96.4%), are shift-reduce errors (equally distributed between shift-becomes-reduce and reduce-becomes-shift), and only 7 (3.6%) are due to reduce-reduce conflicts.[8] These statistics confirm our intuition that shift-reduce decisions are much harder to make during parsing, and contribute to the overwhelming majority of errors, which is studied in the next hypothesis.

Hypothesis 2 is verified in Table 7. We take the gold-standard shift-reduce sequence on the English dev set, and classify them into the four categories based on bilingual contiguity features: (a) $c(s_{t-1}, s_t)$, i.e. whether the top 2 spans on stack is contiguous, and (b) $c_{\text{R}}(s_t, w_i)$, i.e. whether the stack top is contiguous with the current word w_i. According to discussions in Section 3, when (a) is contiguous and (b) is not, it is a clear signal for reduce (to combine the top two elements on the stack) rather than shift, and is strongly supported by the data (first line: 1209 reduces vs. 172 shifts); and while when (b) is contiguous and (a) is not, it should suggest shift (combining s_t and w_i before s_{t-1} and s_t are combined) rather than reduce, and is mildly supported by the data (second line: 1432 shifts vs. 805 reduces). When (a) and (b) are both contiguous or both discontiguous, it should be considered a neutral signal, and is also consistent with the data (next two lines). So to conclude, this bilingual hypothesis is empirically justified.

$c(s_{t-1}, s_t)$	$c_{\text{R}}(s_t, w_i)$	shift		reduce
$+$	$-$	172	$\ll$	1,209
$-$	$+$	1,432	$>$	805
$+$	$+$	4,430	$\sim$	3,696
$-$	$-$	525	$\sim$	576
total		6,559	$=$	6,286

Table 7: [Hyp. 2] Correlation of *gold-standard* shift/reduce decisions with bilingual contiguity conditions (on English dev set). Note there is always one more shift than reduce in each sentence.

On the other hand, we would like to note that these correlations are done with *automatic* word alignments (in our case, from the Berkeley aligner) which can be quite noisy. We suspect (and will finish in the future work) that using *manual* alignments would result in a better correlation, though for the main parsing results (see below) we can only afford automatic alignments in order for our approach to be widely applicable to *any* bitext.

5.4 Results

We incorporate the three bilingual features (again, with automatic alignments) into the baseline parser, retrain it, and test its performance on the English dev set, with varying beam size. Table 8 shows that bilingual constraints help more with larger beams, from almost no improvement with the deterministic mode (k=1) to +0.5% better with the largest beam (k=16). This could be explained by the fact that beam-search is more robust than the deterministic mode, where in the latter, if our

[7] to be really precise one can define "*independent mistakes*" as those not affected by previous ones, i.e., errors made after the parser *recovers* from previous mistakes; but this is much more involved and we leave it to future work.

[8] Note that shift-reduce errors include those due to the non-uniqueness of oracle, i.e., between some reduce_{L} and shift. Currently we are unable to identify "genuine" errors that would result in an incorrect parse. See also Section 2.5.

	baseline		+bilingual	
k	accuracy	time (s)	accuracy	time (s)
1	84.58	0.011	84.67	0.012
2	85.30	0.025	85.62	0.028
4	85.42	0.040	85.81	0.044
8	85.50	0.081	85.95	0.085
16	85.57	0.158	86.07	0.168

Table 8: Effects of beam size k on efficiency and accuracy (on English dev set). Time is average per sentence (in secs). Bilingual constraints show more improvement with larger beams, with a fractional efficiency overhead over the baseline.

	English	Chinese
monolingual baseline	86.9	85.7
+bilingual features	**87.5**	**86.3**
improvement	+0.6	+0.6
signficance level	$p < 0.05$	$p < 0.08$
Berkeley parser	86.1	87.9

Table 9: Final results of dependency accuracy (%) on the test set (290 sentences, beam size k=16).

bilingual features misled the parser into a mistake, there is no chance of getting back, while in the former multiple configurations are being pursued in parallel. In terms of speed, both parsers run proportionally slower with larger beams, as the time complexity is linear to the beam-size. Computing the bilingual features further slows it down, but only fractionally so (just 1.06 times as slow as the baseline at k=16), which is appealing in practice. By contrast, Burkett and Klein (2008) reported their approach of "monolingual k-best parsing followed by bilingual k^2-best reranking" to be "3.8 times slower" than monolingual parsing.

Our final results on the test set (290 sentences) are summarized in Table 9. On both English and Chinese, the addition of bilingual features improves dependency arc accuracies by +0.6%, which is mildly significant using the Z-test of Collins et al. (2005). We also compare our results against the Berkeley parser (Petrov and Klein, 2007) as a reference system, with the exact same setting (i.e., trained on the bilingual data, and testing using gold-standard POS tags), and the resulting trees are converted into dependency via the same headrules. We use 5 iterations of split-merge grammar induction as the 6th iteration overfits the small training set. The result is worse than our baseline on English, but better than our bilingual parser on Chinese. The discrepancy between English and Chinese is probably due to the fact that our baseline feature templates (Table 2) are engineered on English not Chinese.

6 Conclusion and Future Work

We have presented a novel parsing paradigm, *bilingually-constrained monolingual parsing*, which is much simpler than joint (bi-)parsing, yet still yields mild improvements in parsing accuracy in our preliminary experiments. Specifically, we showed a simple method of incorporating alignment features as soft evidence on top of a state-of-the-art shift-reduce dependency parser, which helped better resolve shift-reduce conflicts with fractional efficiency overhead.

The fact that we managed to do this with only three alignment features is on one hand encouraging, but on the other hand leaving the bilingual feature space largely unexplored. So we will engineer more such features, especially with lexicalization and soft alignments (Liang et al., 2006), and study the impact of alignment quality on parsing improvement. From a linguistics point of view, we would like to see how *linguistics distance* affects this approach, e.g., we suspect English-French would not help each other as much as English-Chinese do; and it would be very interesting to see what types of syntactic ambiguities can be resolved across different language pairs. Furthermore, we believe this bilingual-monolingual approach can easily transfer to shift-reduce constituency parsing (Sagae and Lavie, 2006).

Acknowledgments

We thank the anonymous reviewers for pointing to us references about "arc-standard". We also thank Aravind Joshi and Mitch Marcus for insights on PP attachment, Joakim Nivre for discussions on arc-eager, Yang Liu for suggestion to look at manual alignments, and David A. Smith for sending us his paper. The second and third authors were supported by National Natural Science Foundation of China, Contracts 60603095 and 60736014, and 863 State Key Project No. 2006AA010108.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*, volume I: Parsing of *Series in Automatic Computation*. Prentice Hall, Englewood Cliffs, New Jersey.

Ann Bies, Martha Palmer, Justin Mott, and Colin Warner. 2007. English chinese translation treebank v1.0. LDC2007T02.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531–540, Ann Arbor, Michigan, June.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of ACL (poster)*, pages 205–208.

Victoria Fossum and Kevin Knight. 2008. Using bilingual chinese-english word alignments to resolve pp-attachment ambiguity in english. In *Proceedings of AMTA Student Workshop*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*, pages 273–280.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of ACL-IJCNLP*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the ACL: HLT*, Columbus, OH, June.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of COLING-ACL*, Sydney, Australia, July.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd ACL*.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP*, Honolulu, Haiwaii.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of english text. In *Proceedings of COLING*, Geneva.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Incremental Parsing: Bringing Engineering and Cognition Together. Workshop at ACL-2004*, Barcelona.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL*.

Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of ACL (poster)*.

Lee Schwartz, Takako Aikawa, and Chris Quirk. 2003. Disambiguation of english pp attachment using multilingual aligned data. In *Proceedings of MT Summit IX*.

David A. Smith and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous features. In *Proceedings of EMNLP*.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using english to parse korean. In *Proceedings of EMNLP*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *Proceedings of COLING*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of EMNLP*.

Accurate Semantic Class Classifier for Coreference Resolution

Zhiheng Huang[1], **Guangping Zeng**[1,2], **Weiqun Xu**[3], **and Asli Celikyilmaz**[1]
[1]EECS Department, University of California at Berkeley,
CA 94720, USA
{zhiheng,gpzeng,asli}@eecs.berkeley.edu
[2]Computer Science Department, School of Information Engineering,
University of Science and Technology Beijing, China
[3]ThinkIT, Institute of Acoustics, Chinese Academy of Sciences,
Beijing, 100190, China
xuweiqun@hccl.ioa.ac.cn

Abstract

There have been considerable attempts to incorporate semantic knowledge into coreference resolution systems: different knowledge sources such as WordNet and Wikipedia have been used to boost the performance. In this paper, we propose new ways to extract WordNet feature. This feature, along with other features such as named entity feature, can be used to build an accurate *semantic class* (SC) classifier. In addition, we analyze the SC classification errors and propose to use relaxed SC agreement features. The proposed accurate SC classifier and the relaxation of SC agreement features on ACE2 coreference evaluation can boost our baseline system by 10.4% and 9.7% using MUC score and anaphor accuracy respectively.

1 Introduction

Coreference resolution is used to determine which noun phrases (including pronouns, proper names, and common nouns) refer to the same entities in documents. Much work on coreference resolution is based on (Soon et al., 2001), which built a decision tree classifier to label pairs of mentions as coreferent or not. Recent work aims to improve the performance from two aspects: new models and new features. The former cast the pair wise mention classifications into various forms such as the best path in a Bell tree (Luo et al., 2004), the best graph cut (Nicolae and Nicolae, 2006), integer linear programming (Denis and Baldridge, 2007) and graph partition based conditional model (McCallum and Wellner, 2004). The latter develop and investigate new linguistic features for the problem. For instance, WordNet (Poesio et al., 2004), Wikipedia (Ponzetto and Strube, 2006), semantic neighbor words (Ng, 2007a), and pattern based features (Yang and Su, 2007) have been extensively studied.

Deeper linguistic knowledge is required to enable the coreference resolution to reach a higher level of performance (Kehler et al., 2004). An important type of semantic knowledge that has been employed in coreference resolution system is the semantic class (SC) of an NP, which can be used to filter out the coreference between semantically incompatible NPs. However, the difficulty is to accurately compute the semantic class features. In this paper, we show that the WordNet may not be efficiently employed in the traditional way such as (Soon et al., 2001; Ng, 2007a; Ponzetto and Strube, 2006) to compute the semantic class features. We introduce new ways to use the WordNet and the experiments show its effectiveness in determining the semantic classes for noun phrases. In addition, we analyze the classification errors of the SC classifier and propose to use relaxed SC agreement features. With these proposed features and other standard syntactic features (which are commonly employed in existing coreference systems), our coreference resolution system can obtain an increase of 10.4% for MUC score and 9.7% for anaphor accuracy from the baseline in ACE2 evaluation.

2 Related Work

WordNet (Fellbaum, 1998) as an important knowledge source has been widely employed in previous coreference resolution work. For example, Harabagiu et al. (2001) have used WordNet relations such as synonym and is-a to mine the patterns of WordNet paths for pairs of antecedents and anaphors. Due to the nature of the rule based coreference system (in contrast to machine learning based), the weights of relations may not be accurately estimated. Vieira and Poesio (2000) and Markert and Nissim (2005) have used WordNet synonym and hyponym etc. to determine if an anaphor semantically relates to one previous NP. Ponzetto and Strube (2006) have used WordNet semantic similarity and relatedness scores between antecedents and candidate anaphors. Their

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1232–1240,
Singapore, 6-7 August 2009.

work is different to this work in the following: 1) Their work involves various relations such as hyponyms and meronyms while ours only makes use of hypernyms; and 2) Their work focuses on investigating if two NPs have particular WordNet relations or not, while ours focuses on using WordNet hypernyms for their SC classification and then testing their SC compatibility. In doing so, we can directly model the accuracy of semantic class classification and test its impact on coreference resolution.

While the SC of a proper name is computed fairly accurately using a named entity (NE) recognizer, many coreference resolution systems simply assign to a common noun the first (i.e., most frequent) WordNet synset as its SC (Soon et al., 2001; Markert and Nissim, 2005). This heuristics, apparently, did not lead to good performance. The best reported ACE2 coreference resolution system (Ng, 2007a; Ng, 2007b) has proposed an accurate SC classifier which used heterogeneous semantic knowledge sources. WordNet is just one of the several knowledge sources which have been utilized. However, the WordNet based features is not informative compared to other features such as the semantic neighbor feature. Similarly, Ponzetto and Strube (2006) have discovered that the WordNet feature is no more informative than the community-generated Wikipedia feature. In this paper, we focus on the investigation of various usages of WordNet for the SC classification task. The work which is directly comparable to ours would be (Ng, 2007a; Ng, 2007b).

Other similar work includes the *mention detection* (MD) task (Florian et al., 2006) and joint probabilistic model of coreference (Daumé III and Marcu, 2005). The MD task identifies the boundary of a mention, its mention type (e.g., pronoun, name), and its semantic type (e.g., person, organization). Unlike them, we do not perform the boundary detection, as we make use of the noun phrases directly from the noun phrase chunker and NE recognizer. The joint probabilistic model models the MD and coreference simultaneously, while our work focuses on them separately.

3 Semantic Class Classification

In this section, we describe how we compile the training corpus and extract features using WordNet. We report our results on the ACE coreference corpus due to that it has been commonly used and it was annotated SCs of six types.[1] As in (Ng, 2007a), we first train a classifier to predict the SC of an NP. This SC information is used later in the coreference resolution stage. For example, *the audience* is classified as SC of *person*, and it thus should not be coreferent with *the security industry*, which is usually classified as *organization*. This task is by no means trivial. First, while the classification of *Tom Hanks* being SC of *person* can be accurately achieved by an NE recognizer, the association of *audience* and *person* requires semantic language source such as WordNet. Second, the same noun phrase can be annotated with different SCs under different context. For example, *the authorities* is usually annotated as *person*, but it is sometimes as *organization*. Even worse, the same noun phrases are sometimes annotated with one of the five explicitly annotated classes while sometimes are not annotated at all (thus falling into the other SC). For example, *people* is annotated as *person* SC explicitly 20 times and is not annotated at all 21 times in the ACE2 testset. This inconsistent annotation adversely affects the performance of an SC classifier. And this in turn would cause errors during coreference stage. In section 4.3, we show how to relax the strict SC agreement feature to address this.

3.1 Training instance creation

We use ACE Phase 2 Coreference corpus to train the SC classifier. Each noun phrase which is identified by the noun phrase chunker or NE recognizer is used to create a training instance. Each instance is represented by a set of lexical, syntactic and semantic features, as described below. If the NP under consideration is annotated as one of the five ACE SCs in the corpus, then the classification of the associated training instance is the ACE SC of the NP. Otherwise, the instance is labeled as other.

ACE 2 corpus has a training set and a test set which comprise of 422 and 97 texts respectively. We divide the training set into a new training and a development set: the former consists of 90% randomly generated and stratified original training instances and the latter consists of the rest 10% instances. The test set remains the same as in ACE2 corpus. The size of each dataset and its SC distributions are shown in Table 1. Note that the training and development datasets have exactly the same distributions of SCs due to the stratification procedure. That is, each class has the same proportion in training and development datasets. We tune the feature parameters against development set and report performance on both development set and test set.

[1] Person, organization, gpe, location and facility are explicitly annotated. The rest noun phrases are other type.

Table 1: Distributions of SCs in ACE2 corpus.

	Size	PER	ORG	GPE	FAC	LOC	OTH
Train	55629	20.29	7.30	8.42	0.61	0.55	62.80
Dev	6181	20.29	7.30	8.42	0.61	0.55	62.80
Test	15360	20.48	7.57	6.90	0.85	0.41	63.79

3.2 Lexical features

Each instance is represented as a bag of features and is fed into a classifier in training stage. We present four binary lexical feature sets as follows.

Word unigrams and bigrams: An N-gram is a sub-sequence of N words from a given noun phrase. Unigram forms the bag of words feature, and bigram forms the pairs of words feature, and so forth. We have considered word unigram and bigram features in our experiments.

First and last words: This feature extracts the first and last words of an NP. For example, the first word *the* and the last word *store* are extracted from the NP *the main store*. This feature does not only coarsely models the influence of the first word, for example, *a* or *the*, but also models the head word, since the head word usually is the last word in the NP.

Head word: We use Collins style rules (Collins, 1999) to extract the head words for given NPs. These features should be most informative if the training corpus is large enough.[2] For example, the head word *company* of the NP *the company* immediately determines its SC being *organization*. However, due to the sparseness of training data, its potential importance is adversely affected.

3.3 Semantic features

NE feature is extracted from Stanford named entity recognizer (NER) (Finkel et al., 2005). Three types of named entities: person, location and organization can be recognized for a given NP. This feature is primarily useful for SC classification of proper nouns.

WordNet is a large English lexicon in which semantically related words are connected via cognitive synonyms (synsets). The WordNet is a useful tool for word semantics analysis and has been widely used in natural language processing applications. In WordNet, synsets are organized into hierarchies with hypernym/hyponym relationships: Y is a hypernym of X if every X is a (kind of) Y (X is called a hyponym of Y in this case).

The WordNet is employed in (Ng, 2007a) as following to create the **WN_CLASS** feature. For each keyword w as shown in the right column of Table 2, if the head noun of a given NP is a hyponym of w in WordNet,[3] then the word w becomes a feature for such NP. It is explained that these keywords are correlated with the ACE SCs and they are obtained via experimentation with WordNet and the ACE SCs of the NPs in the ACE training data. However, it is likely that these handcrafted keywords have poor coverage for general cases. As a result, it may not make full use of WordNet semantic knowledge. This will be shown in our individual feature contribution experiment in Section 3.5.

Table 2: List of keywords used in WordNet semantic feature in (Ng, 2007a).

ACE SC	Keywords
PER	person
ORG	social group
FAC	establishment, construction, building, facility, workplace
GPE	country, province, government, town, city, administration, society, island, community
LOC	dry land, region, landmass, body of water geographical area, geological formation

There are other ways of using WordNet for semantic feature extraction. For example, Ponzetto and Strube (2006) have employed WordNet similarity measure for coreference resolution. The difference is that they created the feature directly at the coreference resolution stage, ie, using the WordNet similarity between the antecedent and anaphor to determine if they are coreferent, while we focus on using this feature to classify an NP into a particular SC. For comparison, we implemented a WordNet similarity based feature (**WN_SIM**) as follows: for a given NP head word and a key word as listed in Table 2, the WordNet similarity package (Seco et al., 2004) models the length of path traveling from the head word to the key word over the WordNet network. It then computes the semantic similarity based on the path. For example, the similarity between *company* and *social group* is 0.77, while the similarity between *company* and *person* is 0.59. The key word which receives the highest similarity to the head word is marked as a feature.

The WN_CLASS feature may suffer from the coverage problem and the WN_SIM feature is heavily dependent on the definition of similarity metric which may turn out to be inappropriate for coreference resolution task. To make better use of WordNet knowledge, we attempt to directly introduce hypernyms for the NP head words (we denote

[2] It, however, is mostly useful for nominal noun phrase and not for the pronoun and proper noun phrases.

[3] Only the first synset of the NP is used.

it as **WN_HYP** feature). The most similar work to ours is (Daumé III and Marcu, 2005), in which two most common synsets from WordNet for *all* words in an NP and their hypernyms are extracted as features. We avoid augmenting the hypernyms for non-head words in the NP to prevent introducing noisy information, which may potentially corrupt the hypernym feature space.

Considering a WordNet hypernym structure as shown in Fig. 1 for the word *company*, its first synset (an institution created to conduct business) has a unique id of 08058098 and can also be represented by a set of description words (company in this case). Its third synset (the state of being with someone) has an id of 13929588 and description words of company, companionship, fellowship, society. Each synset can be extended by its hypernym synsets. For example, the direct hypernym of the first synset is the synset of 08053576 which can be described as institution, establishment. The augmentation of hypernyms for NP head words can introduce useful information, but can also bring noise if the head word or the synset of head word are not correctly identified. For an optimal use of WordNet hypernyms, four questions shall be addressed: 1) how many depths are required to tradeoff the generality (thus more informative) and the specificity (thus less noisy)? 2) which synset of the given word is needed to be augmented? 3) which representation (synset id or synset word) is better? and 4) is it helpful to encode the hypernym depth into the hypernym feature?[4] These four questions provide the guideline to search the optimal use of WordNet. We will design experiments in Section 3.5 to determine the optimal configuration of WN_HYP feature.

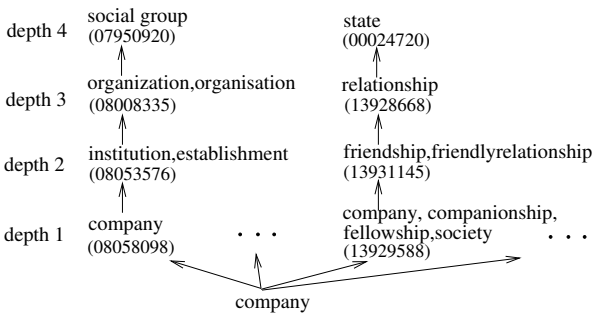


Figure 1: WordNet hypernym hierarchy for the word *company*.

[4]For example, we encode the synset 08053576 as 08053576-1, with the last digit 1 indicating the depth of hypernym with regard to the entry word *company*.

3.4 Learning algorithm

Maximum entropy (ME) models (Berger et al., 1996; Manning and Klein, 2003), also known as log-linear and exponential learning models, has been adopted in the SC classification task. Maximum entropy models can integrate features from many heterogeneous information sources for classification. Each feature corresponds to a constraint on the model. Given a training set of (C, D), where C is a set of class labels and D is a set of feature represented data points, the maximum entropy model attempts to maximize the log likelihood

$$\log P(C|D,\lambda) = \sum_{(c,d)\in(C,D)} \log \frac{\exp \sum_i \lambda_i f_i(c,d)}{\sum_{c'} \exp \sum_j \lambda_j f_i(c,d)} \quad (1)$$

where $f_i(c,d)$ are feature indicator functions and λ_i are the parameters to be estimated. We use ME models for both SC classification and mention pair classification.

3.5 SC classification evaluation

We design three experiments to test the accuracy of our classifiers. The first experiment evaluates the individual contribution of different feature sets to SC classification accuracy. In particular, a ME model is trained on the 55,629 training instances using the following feature sets separately: 1) unigram, 2) bigram, 3) first-last word, 4) head word (HW), 5) named entities (NE), 6) HW+WN_CLASS, 7) HW+WN_SIM, and 8) variants of HW+WN_HYP. Note that HW+WN_CLASS is the semantic feature used in (Ng, 2007a), HW+WN_SIM is the semantic feature using WordNet similarity measure (Seco et al., 2004), and variants of HW+WN_HYP are the work proposed in this paper. We combine head word and the semantic features due to the fact that WordNet features are dependent on head words and they could be treated as units. In the second experiment, features are fed into the ME model incrementally until all features have been used.[5] Finally, we perform the feature ablation experiments. That is, we remove one feature at a time from the entire feature set and test the accuracy loss. The SC classification performance is measured by accuracy, i.e., the proportion of the correctly classified instances among all test instances.

Individual feature contribution Table 3 shows the SC classification accuracy of all NPs (all) and non-pronoun NPs (non-PN) on the development and test datasets using individual feature

[5]The optimal of HW+WN_HYP configuration is used.

sets. Among all the lexical features, unigram feature performs the best (81.3%) for all NPs over the development dataset. The bigram feature performs poorly due to the sparsity problem: NPs usually consist of one to three words. The first-last word feature effectively models the prefix words (such as *a* and *the*) and the head words and thus obtains a reasonably high accuracy of 80.1%. As mentioned before, the head word feature may suffer from the sparsity and it results in the accuracy of 78.2%. We also list the accuracies for non-pronoun NP SC classification, which are slightly different compared to all NP SC classification except for bigram, in which the accuracy has increased 3.9%.

Table 3: SC classification accuracy of ME using individual feature sets for development and test ACE2 datasets.

Feature type	dev		test	
	all	non-PN	all	non-PN
unigram	81.3	81.6	72.4	71.9
bigram	32.5	36.4	26.3	28.4
first-last word	80.1	80.2	71.6	71.0
HW	78.2	78.0	68.3	67.1
NE	74.0	82.8	73.1	81.9
HW+WN_CLASS	79.5	79.4	70.3	69.5
HW+WN_SIM	81.2	81.4	73.8	73.6
HW+WN_HYP (1)	82.6	83.1	74.8	74.7
HW+WN_HYP (3)	82.8	83.4	75.2	75.2
HW+WN_HYP (6)	**83.1**	83.7	75.6	75.7
HW+WN_HYP (9)	83.0	83.6	75.7	75.7
HW+WN_HYP (∞)	83.1	83.7	75.8	75.9
HW+WN_HYP (6) word form	82.8	83.3	75.6	75.7
HW+WN_HYP (6) depth encoded	82.9	83.5	75.4	75.4
HW+WN_HYP (6) first synset	83.0	83.6	76.4	76.6

Although Stanford NER performs well on named entity recognition task, it results in accuracy of 74.0% for all NP SC classification, due to its inability to deal with pronouns such as *he* and common nouns such as *the government*. The removal of pronouns significantly boosts its accuracy to 82.8%. The introduction of semantic feature HW+WN_CLASS boosts the performance to 79.5% compared to the head word alone of 78.2%. This conforms to (Ng, 2007a) that only small gain can be achieved using WN_CLASS feature. The HW+WN_SIM feature outperforms HW+WN_CLASS and the accuracy reaches 81.2%. For the variants of HW+WN_HYP, we first search the optimal depth. This is performed by using all synsets for NP head word, encoding the feature using synset id (rather than synset word), and no hypernym depth is encoded in the features. We try various depths of 1, 3, 6, 9 and ∞, with ∞ signifies that no depth constraint is imposed. The optimal depth of 6 is obtained with the accuracy of 83.1% over the development dataset. We then fix the depth of 6 to try using synset word as features, using synset id with depth encoded as features, and using first synset only. The results show that the optimum is to encode the features using hypernym synset id without hypernym depth information and all synsets are considered for hypernym extraction. This is slightly different from the previous finding (Soon et al., 2001; Lin, 1998b) that a coreference resolution system employing only the first WordNet synset performs slightly better than that employing more than one synset.[6] The best result reaches the accuracy of 83.1%. Although the best semantic feature only outperforms the best lexical feature by 1.8% on the development dataset, its gain in the test dataset is more significant (3.2%, from 72.4% to 75.6%).

Incremental feature contribution Once we use the training and development datasets to find the optimal configuration of HW+WN_HYP semantic feature, we use all lexical features and the optimal HW+WN_HYP feature incrementally to train an ME model over the combination of training and development datasets. Table 4 shows the SC classification accuracy of all NPs (all) and non-pronoun NPs (non-PN) on the training+development (we refer it as training hereafter) and test datasets.

Table 4: SC classification accuracy of ME using incremental feature sets for training and test ACE2 datasets.

Feature type	train		test	
	all	non-PN	all	non-PN
HW	87.8	89.0	68.6	67.6
+WN_HYP	87.8	89.0	75.7	75.8
+unigram	91.5	93.3	77.7	78.1
+bigram	93.1	95.2	78.7	79.2
+first-last word	93.2	95.3	78.8	79.3
+NE	93.4	95.6	**83.1**	**84.4**
Ng 2007a	-	85.0	-	83.3

Note that the significant higher accuracies in training compared to test are due to the overfitting problem. The interesting evaluation thus remains on the test data. As can be seen, the inclusion of more features results in higher performance. This is more obvious in the test dataset than in the training dataset. The inclusion of the

[6] In fact, the accuracy of the test data supports their claims. The accuracy using the first synset compared to using all synsets results in the accuracy increase from 75.6% to 76.4% for all NPs over the test dataset.

optimized WN_HYP feature (ie, using all synsets' hypernyms up to 6 depth and with synset id encoding) results in 7.1% increase for all NP SC classification over test data. This shows the effectiveness of the WN_HYP features to overcome the sparsity of head word feature. The unigram, bigram and first-last word features offer reasonable accuracy gain, and the final inclusion of NE boosts the overall performance to 83.1% for all NP and 84.4% for non pronoun NPs over test data. This result can be directly compared to the SC classification accuracy as reported in (Ng, 2007a), in which the highest accuracy is 83.3% for non pronoun NPs.[7] The large difference between the highest training accuracies is due to that our classifier is trained directly on the ACE2 training dataset, while their SC classifier was trained on BBN Entity Type Corpus (Weischedel and Brunstein, 2005), which is five times larger than the ACE2 corpus used by us. In addition to WordNet, they have adopted multiple knowledge sources which include BBN's IdentiFinder (this is equivalent to the Stanford NER in our work), BLLIP corpus and Reuters Corpus,[8] and dependency based thesaurus (Lin, 1998a). It is remarkable that our SC classifier can achieve even higher accuracy only using WordNet hypernym and NE features. It is worth noting that the small accuracy gain is indeed hard to achieve considering that the test data size is large (15360).

Feature ablation experiment We now perform the feature ablation experiments to further determine the importance of individual features. We remove one feature at a time from the entire feature set. Table 5 shows the SC classification accuracy of all NPs (all) and non-pronoun NPs (non-PN) on the training and test datasets respectively.

Table 5: SC classification accuracy of ME by removing one feature at a time for training and test ACE2 datasets.

Feature type	train		test	
	all	non-PN	all	non-PN
overall	93.4	95.6	83.1	84.4
-HW	93.4	95.5	82.9	84.2
-WN_HYP	93.4	95.5	82.6	83.8
-HW+WN_HYP	93.4	95.5	82.3	83.5
-unigram	93.4	95.5	82.9	84.2
-bigram	92.5	94.5	82.7	84.0
-first-last word	93.4	95.5	82.9	84.1
-NE	93.2	95.3	78.8	79.3

Again, the significant higher accuracies in training compared to test are due to overfitting. The removal of NE feature results in the largest accuracy loss of 4.3% (from 83.1% to 78.8%) for all nouns on test data. It follows WN_HYP (0.5% loss) and the bigram (0.4%). If we treat HW+WN_HYP as one feature, the removal of it results in accuracy loss of 0.8% for all nouns on test data. The unigram, first-last word and head word each results in the loss of 0.2%. The reason that the removal of NE results in a much significant loss is due to the fact that the NE feature is quite different from other features. Its strength is to distinguish SCs for proper names, while other features are more similar (their targets are common nouns). The proposed use of HW+WN_HYP can bring 0.8% gain on top of other features, higher than other informative lexical features including unigram and first-last word.

3.6 Error analysis

A closer look at the errors produced by our SC classifier reveals that the second probable label is very likely to be the actual labels if the first probable one is wrong. In fact, if we allow the classifier to predict two most probable labels and the classification is judged to be true if the actual label is one of the two predictions, then the classification accuracy increases from 83.1% to 96.4%. This is because that the same noun phrases are sometimes annotated with one of the five explicitly annotated classes while sometimes are not annotated at all (thus falling into the other SC). Again for the example of *people*. It is annotated as *person* SC 20 times and is not annotated at all 21 times. Given the same feature set for this instance, the best the classifier can do is to classify it to *other* semantic class. To address this annotation inconsistency issue, we relax the SC agreement feature from the strict match in designing coreference resolution features. For example, if the first probable SC of an NP matches the second probable SC of another NP, we still give some partial match credit.

4 Application to Coreference Resolution

We can now incorporate the NP SC classifier into our ME based coreference resolution system. This section examines how our WordNet hypernym features help improve the coreference resolution performance.

4.1 Experimental setup

We use the ACE-2 (version 1.0) coreference corpus. Each raw text in this corpus was preprocessed automatically by a pipeline of NLP components, including sentence boundary detection,

[7] All NP accuracy was not reported as they excluded the pronouns in creating their training and test data.

[8] They use these corpus to extract patterns to induce SC of common nouns.

POS-tagging and text chunking. The statistics of corpus and mention extraction are shown in Table 6, where g-mention is the automatically extracted mentions which contain the annotated (gold) mentions. The recalls of gold mentions are 95.88% and 95.93% for training and test data respectively.

Table 6: Statistics for corpus and extracted mentions.

	text#	mention#	g-mention#	gold#	recall(%)
train	422	61810	22990	23977	95.88
test	97	15360	5561	5797	95.93

Our coreference system uses Maximum Entropy model to determine whether two NPs are coreferent. As in (Soon et al., 2001; Ponzetto and Strube, 2006), we generate training instances as follows: a positive instance is created for each anaphoric NP, NP_j, and its closest antecedent, NP_i; and a negative instance is created for NP_j paired with each of the intervening NPs, NP_{i+1}, NP_{i+2}, ..., NP_{j-1}. Each instance is represented by syntactic or semantic features described as follows. All training data are used to train a maximum entropy model. In the test stage,we select the closest preceding NP that is classified as coreferent with NP_j as the antecedent of NP_j. If no such NP exists, no antecedent is selected for NP_j.

Unlike other natural language processing tasks such as information extraction which have de facto evaluation metrics, it is an open question which evaluation is the most suitable one. The evaluation becomes more complicated when automatically extracted mentions (in contrast to the gold mentions) are used. To facilitate the comparison with previous work, we report performance using two different scoring metrics: the commonly-used MUC scorer (Vilain et al., 1995) and the *accuracy* of the anaphoric references (Ponzetto and Strube, 2006). An anaphoric reference is correctly resolved if it and its closest antecedent are in the same coreference chain in the resulting partition.

4.2 Baseline features

We briefly review the baseline features used in this paper as follows. More detailed information and implementations can be found at (Soon et al., 2001; Versley et al., 2008). For example, the ALIAS feature takes values of true or false. The value of true means that the antecedent and the anaphor refer to the same entity (date, person, organization or location). The ALIAS feature detection works differently depending on the named entity type. For date, the day, month, and year values are extracted and compared. For person, the last words of the noun phrases are compared. For organization names, the alias detection checks for acronym match such as *IBM* and *International Business Machines Corp*.

Lexical features STRING MATCH: true if NP_i and NP_j have the same spelling after removing article and demonstrative pronouns, false otherwise. ALIAS: true if NP_j is the alias of NP_i.

Grammatical features I_PRONOUN: true if NP_i is a pronoun; J_PRONOUN: true if NP_j is pronoun; J_REFL_PRONOUN: true if NP_j is reflexive pronoun; J_PERS_PRONOUN: true if NP_j is personal pronoun; J_POSS_PRONOUN: true if NP_j is possessive pronoun; J_PN: true if NP_j is proper noun; J_DEF: true if NP_j starts with *the*; J_DEM: true if NP_j starts with *this*, *that*, *these* or *those*; J_DEM_NOMINAL: true if NP_j is a demonstrative nominal noun; J_DEM_PRONOUN: true if NP_j is a demonstrative pronoun; PROPER_NAME: true if both NP_i and NP_j are proper names; NUMBER: true if NP_i and NP_j agree in number; GENDER: true if NP_i and NP_j agree in gender; APPOSITIVE: true if NP_i and NP_j are appositions.

Distance feature DISTANCE: how many sentences NP_i and NP_j are apart.

Semantic feature SEMCLASS: This feature is implemented from (Soon et al., 2001). Its possible values are true, false, or unknown. First the following semantic classes are defined: *female*, *male*, *person*, *organization*, *location*, *date*, *time*, *money*, *percent*, and *object*. Each of these defined semantic classes is then mapped to a WordNet synset. Then the semantic class determination module determines the semantic class for every NP as the first synset of the head noun of the NP. If such synset is a hyponym of defined semantic class, then such semantic class is assigned to the NP. Otherwise, *unknown* class is assigned. Finally, the agreement of semantic classes of NP_i and NP_j is unknown if either assigned class is *unknown*; true if their assigned class are the same, false otherwise. Notice that the WordNet use in (Ng, 2007a) and this feature apply in the same principle except that 1) the former is used in SC classification while the latter is used directly for coreference resolution, and 2) they have different semantic class categories.

4.3 Proposed WordNet agreement features

For each instance which consists of NP_i and NP_j, we apply our SC classifier to label them, say l_i and l_j respectively. We then use these two induced la-

bels to propose the SC agreement feature for NP_i and NP_j. In particular, SC_STRICT is true if l_i and l_j are the same and they are not of other type, false otherwise; SC_COARSE is true if both l_i and l_j are not of other type; In addition, we propose two other SC agreement features to cope with the SC classification errors. SC_RELAX1 is true if the first probable of NP_i, l_{i1}, is not other type and is the same as the second probable of N_j, l_{j2}, or vice visa. SC_RELAX2 is true if the second probable of NP_i, l_{i2}, is not other type and is the same as the second probable label of NP_j, l_{j2}. The purpose in using SC_RELAX1 and SC_RELAX2 features is to relax the strict SC agreement feature in the hope that partial SC match is useful for coreference resolution.

4.4 Coreference results

Table 7 shows the MUC score for ACE2 corpus and its three partitions: bnews, npaper, and nwire using baseline and the proposed semantic features. It also shows the accuracy of resolving anaphors for all nouns in ACE2 corpus. SC_STRICT is the configuration that uses the baseline features with the SEMCLASS (Soon et al., 2001) replaced by SC_STRICT, and SC_COARSE, SC_RELAX1, and SC_RELAX2 are incrementally included into the SC_STRICT feature set.

As can be seen, the SC_STRICT significantly boosts the performance: it improves the MUC F score and anaphor accuracy of baseline from 57.7% to 65.7% and 37.7% to 46.3% respectively. It is remarkable that the new use of WordNet can obtain such significant gain in both MUC score and anaphor accuracy. The large improvement of the precision from 58.1% to 73.3% for all NPs shows that the SC_STRICT feature can effectively filter out the semantic incompatible pairs of antecedents and anaphors. In accordance with our hypothesis, the relaxation of strict SC agreement by including SC_COARSE, SC_RELAX1 and SC_RELAX2 help improve the performance further, which is reflected by both MUC score and anaphor accuracy. For example, compared to the baseline, the use of all proposed four SC agreement features results in the maximal accuracy gain of 9.7% (from 37.7% to 47.4%) and the use of SC_STRICT, SC_COARSE, and SC_RELAX1 results in the maximal MUC score gain of 10.4% (from 57.7% to 68.1%).

Our best MUC score is 68.1% which outperforms the MUC score of 64.6% as reported in (Ng, 2007a) by 3.5%, while our best accuracy of anaphor is 47.4%, which is 4.1% less than the accuracy of 51.5% in (Ng, 2007a). Note that, unlike (Ng, 2007a) which performed extensive experiments using different machine learning algorithms, alternative use of features (either constraint or normal features), and heterogeneous knowledge sources, this paper simply uses one learning classifier (ME model) and only employs WordNet and Stanford NER semantic sources.

The different MUC and accuracy scores reflect the non-trivial cases of evaluating coreference systems. While we leave out the discussion of which evaluation is more appropriate, we focus on showing that the proposed SC classifier can bring significant boost from the baseline using both MUC and accuracy metrics.

5 Conclusion

We have showed that the traditional use of WordNet in coreference resolution may not effectively exploit the WordNet semantic knowledge. We proposed new ways to extract WordNet feature. This feature, along with other features such as named entity feature, can be used to build an accurate *semantic class* (SC) classifier. In addition, we analyzed the classification errors of the SC classifier and relaxed SC agreement features to cope with part of the classification errors. The proposed accurate SC classifier and the relaxation of SC agreement features can boost our baseline coreference resolution system by 10.4% and 9.7% using MUC score and anaphor accuracy respectively.

Acknowledgments

We wish to thank Yannick Versley for his support with BART coreference resolution system and the three anonymous reviewers for their invaluable comments. This research was supported by British Telecom grant CT1080028046 and BISC Program of UC Berkeley.

References

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

M. Collins 1999. Head-driven statistical models for natural language parsing. *PhD thesis*, University of Pennsylvania.

H. Daumé III and D. Marcu. 2005. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proc. of HLT/EMNLP*, pages 97-104.

Table 7: MUC score and accuracy of baseline and proposed SC agreement features for ACE2 dataset.

	MUC score												Accuracy
	All			bnews			npaper			nwire			All
System	R	P	F	R	P	F	R	P	F	R	P	F	
baseline	57.4	58.1	57.7	56.6	55.4	56.0	59.3	60.4	59.9	56.2	58.6	57.3	37.7
Ng 2007a	59.5	70.6	64.6	-	-	-	-	-	-	-	-	-	**51.5**
SC_STRICT	59.6	73.3	65.7	61.6	72.8	66.7	60.3	74.9	66.8	56.8	72.1	63.5	46.3
+ SC_COARSE	59.2	76.7	66.8	61.0	76.7	67.9	59.8	77.2	67.4	56.6	76.2	64.9	45.9
+ SC_RELAX1	59.8	79.0	**68.1**	61.3	79.8	69.3	60.9	80.3	69.3	57.2	76.7	65.5	47.2
+ SC_RELAX2	60.2	77.7	67.8	61.5	78.2	68.9	61.4	78.9	69.1	57.5	75.7	65.4	47.4

P. Denis and J. Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *HLT-NAACL*.

C. Fellbaum. 1998. *An electronic lexical database*. The MIT press.

J. Finkel, T. Grenager, and C. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proc. of ACL*, pages 363-370.

R. Florian, H. Jing, N. Kambhatla, and I. Zitouni. 2006. Factorizing complex models: a case study in mention detection. In *Proc. of COLING/ACL*, pages 473-480.

S. M. Harabagiu, R. C. Bunescu, and S. J. Maiorano. 2001. Text and knowledge mining for coreference resolution. In *Proc. of NAACL*, pages 55-62.

A. Kehler, D. Appelt, L. Taylor, and A. Simma. 2004. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proc. of HLT/NAACL*, pages 289-296.

D. Lin. 1998a. Automatic retrieval and clustering of similar words. In *Proc. of COLING/ACL*, pages 768-774.

D. Lin. 1998b. Using collocation statistics in information extraction. In *Proc. of MUC-7*.

X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. 2004. A mention synchronous coreference resolution algorithm based on the Bell tree. In *Proc. of the ACL*.

C. Manning and D. Klein. 2003. Optimization, Maxent Models, and Conditional Estimation without Magic. *Tutorial at HLT-NAACL 2003 and ACL 2003*.

K. Markert and M. Nissim. 2005. Comparing knowledge sources for nominal anaphora resolution. *Computational Linguistics*, 31(3):367-401.

A. McCallum and B. Wellner. 2004. Conditional models of identity uncertainty with application to noun coreference. In *Proc. of the NIPS*.

V. Ng. 2007a. Semantic Class Induction and Coreference Resolution. In *Proc. of the ACL*.

V. Ng. 2007b. Shallow Semantics for Coreference Resolution. In *Proc. of the IJCAI*.

C. Nicolae and G. Nicolae 2006. BESTCUT: A Graph Algorithm for Coreference Resolution. In *Proc. of the EMNLP*.

M. Poesio, R. Mehta, A. Maroudas, and J. Hitzeman. 2004. Learning to resolve bridging references. In *Proc. of the ACL*.

S. P. Ponzetto and M. Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proc. of the HLT/NAACL*, pages 192-199.

N. Seco, T. Veale, and J. Hayes. 2004. An Intrinsic Information Content Metric for Semantic Similarity in WordNet. *Proc. of the European Conference of Artificial Intelligence*.

W. M. Soon, H. T. Ng, and D. C. Y. Lim. A machine learning approach to coreference resolution of noun phrases. *Computation Linguistics*, 27(4):521-544.

Y. Versley, S. P. Ponzetto, M. Poesio, V. Eidelman, A. Jern, J. Smith, X. Yang, and A. Moschitti. 2008. BART: a modular toolkit for coreference resolution. *ACL 2008 System demo*.

R. Vieira and M. Poesio. 2000. An empirically-based system for processing definite descriptions. *Computational Linguistics*, 26(4):539-593.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoreing scheme. In *Proc. of MUC-6*, pages 45-52.

R. Weischedel and A. Brunstein. 2005. BBN pronoun coreference and entity type corpus. Linguistic Data Consortium.

X. Yang and J. Su. 2007. Coreference resolution using semantic relatedness information from automatically discovered pattens. In *Proc. of the ACL*.

Real-Word Spelling Correction using Google Web 1T 3-grams

Aminul Islam
Department of Computer Science
University of Ottawa
Ottawa, ON, K1N 6N5, Canada
mdislam@site.uottawa.ca

Diana Inkpen
Department of Computer Science
University of Ottawa
Ottawa, ON, K1N 6N5, Canada
diana@site.uottawa.ca

Abstract

We present a method for detecting and correcting multiple real-word spelling errors using the Google Web 1T 3-gram data set and a normalized and modified version of the Longest Common Subsequence (LCS) string matching algorithm. Our method is focused mainly on how to improve the detection recall (the fraction of errors correctly detected) and the correction recall (the fraction of errors correctly amended), while keeping the respective precisions (the fraction of detections or amendments that are correct) as high as possible. Evaluation results on a standard data set show that our method outperforms two other methods on the same task.

1 Introduction

Real-word spelling errors are words in a text that occur when a user mistakenly types a correctly spelled word when another was intended. Errors of this type may be caused by the writer's ignorance of the correct spelling of the intended word or by typing mistakes. Such errors generally go unnoticed by most spellcheckers as they deal with words in isolation, accepting them as correct if they are found in the dictionary, and flagging them as errors if they are not. This approach would be sufficient to detect the non-word error *myss* in "It doesn't know what the *myss* is all about." but not the real-word error *muss* in "It doesn't know what the *muss* is all about." To detect the latter, the spell-checker needs to make use of the surrounding context such as, in this case, to recognise that *fuss* is more likely to occur than *muss* in the context of *all about*. Ironically, errors of this type may even be caused by spelling checkers in the correction of non-word spelling errors when the *auto-correct* feature in some word-processing software sometimes silently change a non-word to the wrong real word (Hirst and Budanitsky, 2005), and sometimes when correcting a flagged error, the user accidentally make a wrong selection from the choices offered (Wilcox-O'Hearn et al., 2008).

An extensive review of real-word spelling correction is given in (Pedler, 2007; Hirst and Budanitsky, 2005) and the problem of spelling correction more generally is reviewed in (Kukich, 1992).

The Google Web 1T data set (Brants and Franz, 2006), contributed by Google Inc., contains English word *n*-grams (from unigrams to 5-grams) and their observed frequency counts calculated over 1 trillion words from web page text collected by Google in January 2006. The text was tokenised following the Penn Treebank tokenisation, except that hyphenated words, dates, email addresses and URLs are kept as single tokens. The sentence boundaries are marked with two special tokens <S> and </S>. Words that occurred fewer than 200 times were replaced with the special token <UNK>. Table 1 shows the data sizes of the Web 1T corpus. The n-grams themselves must appear at least 40 times to be included in the Web 1T corpus[1]. It is expected that this data will be useful for statistical language modeling, e.g.,

Table 1: Google Web 1T Data Sizes

Number of	Number	Size on disk (in KB)
Tokens	1,024,908,267,229	N/A
Sentences	95,119,665,584	N/A
Unigrams	13,588,391	185,569
Bigrams	314,843,401	5,213,440
Trigrams	977,069,902	19,978,540
4-grams	1,313,818,354	32,040,884
5-grams	1,176,470,663	33,678,504

[1]Details of the Google Web 1T data set can be found at www.ldc.upenn.edu/Catalog/docs/LDC2006T13/readme.txt

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1241–1249,
Singapore, 6-7 August 2009.

for machine translation or speech recognition, as well as for other uses.

In this paper, we present a method for detecting and correcting multiple real-word spelling errors using the Google Web 1T 3-gram data set, and a normalized and modified version of the Longest Common Subsequence (LCS) string matching algorithm (details are in section 3.1). By *multiple* errors, we mean that if we have n words in the input sentence, then we try to detect and correct at most n-1 errors. We do not try to detect and correct an error, if any, in the first word as it is not computationally feasible to search in the Google Web 1T 3-grams while keeping the first word in the 3-gram as a variable. Our intention is to focus on how to improve the detection recall (the fraction of errors correctly detected) or correction recall (the fraction of errors correctly amended) while maintaining the respective precisions (the fraction of detections or amendments that are correct) as high as possible. The reason behind this intention is that if the recall for any method is around 0.5, this means that the method fails to detect or correct around 50 percent of the errors. As a result, we can not completely rely on these type of methods, for that we need some type of human interventions or suggestions to detect or correct the rest of the undetected or uncorrected errors. Thus, if we have a method that can detect or correct almost 80 percent of the errors, even generating some extra candidates that are incorrect is more helpful to the human.

This paper is organized as follow: Section 2 presents a brief overview of the related work. Our proposed method is described in Section 3. Evaluation and experimental results are discussed in Section 4. We conclude in Section 5.

2 Related Work

Work on real-word spelling correction can roughly be classified into two basic categories: methods based on semantic information or human-made lexical resources, and methods based on machine learning or probability information. Our proposed method falls into the latter category.

2.1 Methods Based on Semantic Information

The 'semantic information' approach first proposed by Hirst and St-Onge (1998) and later developed by Hirst and Budanitsky (2005) detected semantic anomalies, but was not restricted to checking words from predefined confusion sets. This approach was based on the observation that the words that a writer intends are generally semantically related to their surrounding words, whereas some types of real-word spelling errors are not, such as (using Hirst and Budanitsky's example), "It is my sincere hole (hope) that you will recover swiftly." Such "malapropisms" cause "a perturbation of the cohesion (and coherence) of a text." Hirst and Budanitsky (2005) use semantic distance measures in WordNet (Miller et al., 1993) to detect words that are potentially anomalous in context - that is, semantically distant from nearby words; if a variation in spelling results in a word that was semantically closer to the context, it is hypothesized that the original word is an error (a "malapropism") and the closer word is its correction.

2.2 Methods Based on Machine Learning

Machine learning methods are regarded as lexical disambiguation tasks and confusion sets are used to model the ambiguity between words. Normally, the machine learning and statistical approaches rely on pre-defined confusion sets, which are sets (usually pairs) of commonly confounded words, such as {*their*, *there*, *they're*} and {*principle*, *principal*}. The methods learn the characteristics of typical context for each member of the set and detect situations in which one member occurs in context that is more typical of another. Such methods, therefore, are inherently limited to a set of common, predefined errors, but such errors can include both content and function words. Given an occurrence of one of its confusion set members, the spellchecker's job is to predict which member of that confusion set is the most appropriate in the context. Golding and Roth (1999), an example of a machine-learning method, combined the Winnow algorithm with weighted-majority voting, using nearby and adjacent words as features. Another example of a machine-learning method is that of Carlson et al. (2001).

2.3 Methods Based on Probability Information

Mays et al. (1991) proposed a statistical method using word-trigram probabilities for detecting and correcting real-word errors without requiring predefined confusion sets. In this method, if the trigram-derived probability of an observed sentence is lower than that of a sentence obtained by replacing one of the words with a spelling varia-

tion, then we hypothesize that the original is an error and the variation is what the user intended.

Wilcox-O'Hearn et al. (2008) analyze the advantages and limitations of Mays et al. (1991)'s method, and present a new evaluation of the algorithm, designed so that the results can be compared with those of other methods, and then construct and evaluate some variations of the algorithm that use fixed-length windows. They consider a variation of the method that optimizes over relatively short, fixed-length windows instead of over a whole sentence (except in the special case when the sentence is smaller than the window), while respecting sentence boundaries as natural breakpoints. To check the spelling of a span of d words requires a window of length d+4 to accommodate all the trigrams that overlap with the words in the span. The smallest possible window is therefore 5 words long, which uses 3 trigrams to optimize only its middle word. They assume that the sentence is bracketed by two BoS and two EoS markers (to accommodate trigrams involving the first two and last two words of the sentence). The window starts with its left-hand edge at the first BoS marker, and the Mays et al. (1991)'s method is run on the words covered by the trigrams that it contains; the window then moves d words to the right and the process repeats until all the words in the sentence have been checked. As Mays et al. (1991)'s algorithm is run separately in each window, potentially changing a word in each, Wilcox-O'Hearn et al. (2008)'s method as a side-effect also permits multiple corrections in a single sentence.

Wilcox-O'Hearn et al. (2008) show that the trigram-based real-word spelling-correction method of Mays et al. (1991) is superior in performance to the WordNet-based method of Hirst and Budanitsky (2005), even on content words ("malapropisms"), especially when supplied with a realistically large trigram model. Wilcox-O'Hearn et al. (2008) state that their attempts to improve the method with smaller windows and with multiple corrections per sentence were not successful, because of excessive false positives.

Verberne (2002) proposed a trigram-based method for real-word errors without explicitly using probabilities or even localizing the possible error to a specific word. This method simply assumes that any word trigram in the text that is attested in the British National Corpus (Burnard, 2000) is correct, and any unattested trigram is a likely error. When an unattested trigram is observed, the method then tries the spelling variations of all words in the trigram to find attested trigrams to present to the user as possible corrections. The evaluation of this method was carried out on only 7100 words of the Wall Street Journal corpus, with 31 errors introduced (i.e., one error in every approximately 200 words) obtaining a recall of 0.33 for correction, a precision of 0.05 and a F-measure of 0.086.

3 Proposed Method

The proposed method first tries to determine some probable candidates and then finds the best one among the candidates or sorts them based on some weights. We consider a string similarity function and a normalized frequency value function in our method. The following sections present a detailed description of each of these functions followed by the procedure to determine some probable candidates along with the procedure to sort the candidates.

3.1 Similarity between Two Strings

We use the longest common subsequence (LCS) (Allison and Dix, 1986) measure with some normalization and small modifications for our string similarity measure. We use the same three different modified versions of LCS that we (Islam and Inkpen, 2008) used, along with another modified version of LCS, and then take a weighted sum of these[2]. Kondrak (2005) showed that edit distance and the length of the longest common subsequence are special cases of n-gram distance and similarity, respectively. Melamed (1999) normalized LCS by dividing the length of the longest common subsequence by the length of the longer string and called it longest common subsequence ratio (LCSR). But LCSR does not take into account the length of the shorter string which sometimes has a significant impact on the similarity score.

Islam and Inkpen (2008) normalized the *longest common subsequence* so that it takes into account the length of both the shorter and the longer string and called it *normalized longest common subse-*

[2]We (Islam and Inkpen, 2008) use modified versions because in our experiments we obtained better results (precision and recall) for schema matching on a sample of data than when using the original LCS, or other string similarity measures.

quence (NLCS) which is:

$$v_1 = NLCS(s_i, s_j) = \frac{len(LCS(s_i, s_j))^2}{len(s_i) \times len(s_j)} \quad (1)$$

While in classical LCS, the common subsequence needs not be consecutive, in spelling correction, a consecutive common subsequence is important for a high degree of matching. We (Islam and Inkpen, 2008) used *maximal consecutive longest common subsequence* starting at character 1, $MCLCS_1$ and *maximal consecutive longest common subsequence* starting at any character n, $MCLCS_n$. $MCLCS_1$ takes two strings as input and returns the shorter string or maximal consecutive portions of the shorter string that consecutively match with the longer string, where matching must be from first character (character 1) for both strings. $MCLCS_n$ takes two strings as input and returns the shorter string or maximal consecutive portions of the shorter string that consecutively match with the longer string, where matching may start from any character (character n) for both of the strings. We normalized $MCLCS_1$ and $MCLCS_n$ and called it *normalized* $MCLCS_1$ ($NMCLCS_1$) and *normalized* $MCLCS_n$ ($NMCLCS_n$), respectively.

$$v_2 = NMCLCS_1(s_i, s_j) = \frac{len(MCLCS_1(s_i, s_j))^2}{len(s_i) \times len(s_j)} \quad (2)$$

$$v_3 = NMCLCS_n(s_i, s_j) = \frac{len(MCLCS_n(s_i, s_j))^2}{len(s_i) \times len(s_j)} \quad (3)$$

Islam and Inkpen (2008) did not consider consecutive common subsequences ending at the last character, though $MCLCS_n$ sometimes covers this, but not always. We argue that the consecutive common subsequence ending at the last character is as significant as the consecutive common subsequence starting at the first character. So, we introduce the *maximal consecutive longest common subsequence* ending at the last character, $MCLCS_z$ (Algorithm 1). Algorithm 1, takes two strings as input and returns the shorter string or the maximal consecutive portions of the shorter string that consecutively matches with the longer string, where matching must end at the last character for both strings. We normalize $MCLCS_z$ and call it *normalized* $MCLCS_z$ ($NMCLCS_z$).

$$v_4 = NMCLCS_z(s_i, s_j) = \frac{len(MCLCS_z(s_i, s_j))^2}{len(s_i) \times len(s_j)} \quad (4)$$

We take the weighted sum of these individual values v_1, v_2, v_3, and v_4 to determine string similarity score, where α_1, α_2, α_3, α_4 are weights and $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1$. Therefore, the similarity of the two strings, $S \in [0, 1]$ is:

$$S(s_i, s_j) = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + \alpha_4 v_4 \quad (5)$$

We heuristically set equal weights for our experiments[3]. Theoretically, $v_3 \geq v_2$ and $v_3 \geq v_4$. To give an example, consider $s_i = albastru$ and $s_j = alabasteru$, then

$LCS(s_i, s_j) = albastru$
$MCLCS_1(s_i, s_j) = al$
$MCLCS_n(s_i, s_j) = bast$
$MCLCS_z(s_i, s_j) = ru$
$NLCS(s_i, s_j) = 8^2/(8 \times 10) = 0.8$
$NMCLCS_1(s_i, s_j) = 2^2/(8 \times 10) = 0.05$
$NMCLCS_n(s_i, s_j) = 4^2/(8 \times 10) = 0.2$
$NMCLCS_z(s_i, s_j) = 2^2/(8 \times 10) = 0.05$

The string similarity, $S = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + \alpha_4 v_4 = 0.25 \times 0.8 + 0.25 \times 0.05 + 0.25 \times 0.2 + 0.25 \times 0.05 = 0.275$

3.2 Normalized Frequency Value

We determine the normalized frequency value of each candidate word for a single position with respect to all other candidates for the same position. If we find n replacements of a word w_i which are $\{w_{i1}, w_{i2}, \cdots, w_{ij}, \cdots, w_{in}\}$, and their frequencies $\{f_{i1}, f_{i2}, \cdots, f_{ij}, \cdots, f_{in}\}$, where f_{ij} is the frequency of a 3-gram (where any candidate word w_{ij} is a member of the 3-gram), then we determine the normalized frequency value of any candidate word w_{ij}, represented as $F(w_{ij}) \in (0, 1]$, as the frequency of the 3-gram having w_{ij} over the maximum frequency among all the candidate words for that position:

$$F(w_{ij}) = \frac{f_{ij}}{\max(f_{i1}, f_{i2}, \cdots, f_{ij}, \cdots, f_{in})} \quad (6)$$

3.3 Determining Candidate Words

Our task is to correct real-word spelling error from an input text using Google Web 1T 3-gram data set. Let us consider an input text W which

[3]We use equal weights in several places in this paper in order to keep the system unsupervised. If development data would be available, we could adjust the weights.

Algorithm 1: $MCLCS_z$ (Maximal Consecutive LCS ending at the last character)

input : s_i, s_j /* s_i and s_j are input strings where $|s_i| \leq |s_j|$ */
output: str /* str is the Maximal Consecutive LCS ending at the last character */

```
str ← NULL
c ← 1
while |s_i| ≥ c do
    x ← SubStr(s_i, −c, 1)    /* returns cth character of s_i from the end */
    y ← SubStr(s_j, −c, 1)    /* returns cth character of s_j from the end */
    if x = y then
        str ← SubStr(s_i, −c, c)
    else
        return str
    end
    increment c
end
```

after tokenization[4] has m words, i.e., $W = \{w_1, w_2, \ldots, w_m\}$. Our method aims to correct m-1 spelling errors, for all m-1 word positions, except for the first word position, as we do not try to correct the first word. We use a slight different way to correct the first word (i.e., w_2) and the last word (i.e., w_m) among those m-1 words, than for the rest of the words. First, we discuss how we find the candidates for a word (say w_i, where $2<i<m$) which is not either w_2 or w_m. Then, we discuss the procedure to find the candidates for either w_2 or w_m. Our method could have worked for the first word too. We did not do it here due to efficiency reasons. Google 3-grams are sorted based on the first word, then the second word, and so on. Based on this sorting, all Google 3-grams are stored in 97 different files. All the 97 Google 3-gram files could have been needed to access a single word, instead of accessing just one 3-gram file as we do for any other words. This is because when the first word needs to be corrected, it might be in any file among those 97 Google 3-gram files. No error appears in the first position among 1402 inserted malapropisms. The errors start appearing from the second position till the last position.

[4]We need to tokenize the input sentence to make the 3-grams formed using the tokens returned after the tokenization consistent with the Google 3-grams. The input sentence is tokenized in a manner similar to the tokenization of the Wall Street Journal portion of the Penn Treebank. Notable exceptions include the following:

- Hyphenated word are usually separated, and hyphenated numbers usually form one token.
- Sequences of numbers separated by slashes (e.g., in dates) form one token.
- Sequences that look like urls or email addresses form one token.

3.3.1 Determining Candidate Words for w_i $(2 < i < m)$

We use the following steps:

1. We define the term *cut off frequency* for word w_i or word w_{i+1} as the frequency of the 3-gram $w_{i-1}\ w_i\ w_{i+1}$ in the Google Web 1T 3-grams, if the said 3-gram exists. Otherwise, we set the *cut off frequency* of w_i as 0. The intuition behind using the *cut off frequency* is the fact that, if the word is misspelled, then the correct one should have a higher frequency than the misspelled one. Thus, using the *cut off frequency*, we isolate a large number of candidates that we do not need to process.

2. We find all the 3-grams (where only w_i is changed while w_{i-1} and w_{i+1} are unchanged) having frequency greater than the *cut off frequency* of w_i (determined in step 1). Let us consider that we find n replacements of w_i which are $R_1 = \{w_{i1}, w_{i2}, \cdots, w_{in}\}$ and their frequencies $F_1 = \{f_{i1}, f_{i2}, \cdots, f_{in}\}$ where f_{ij} is the frequency of the 3-gram $w_{i-1}\ w_{ij}\ w_{i+1}$.

3. We determine the *cut off frequency* for word w_{i-1} or word w_i as the frequency of the 3-gram $w_{i-2}\ w_{i-1}\ w_i$ in the Google Web 1T 3-grams, if the said 3-gram exists. Otherwise, we set the *cut off frequency* of w_i as 0.

4. We find all the 3-grams (where only w_i is changed while w_{i-2} and w_{i-1} are unchanged) having frequency greater than the *cut off frequency* of w_i (determined in step 3). Let us consider that we find n replacements of w_i which are $R_2 = \{w_{i1}, w_{i2}, \cdots, w_{in}\}$ and their frequencies

$F_2 = \{f_{i1}, f_{i2}, \cdots, f_{in}\}$ where f_{ij} is the frequency of the 3-gram $w_{i-2}\ w_{i-1}\ w_{ij}$.

5. For each $w_{ij} \in R_1$, we calculate the string similarity between w_{ij} and w_i using equation (5) and then assign a weight using the following equation (7) only to the words that return the string similarity value greater than 0.5.

$$weight = \beta S(w_i, w_{ij}) + (1-\beta)F(w_{ij}) \quad (7)$$

Equation (7) is used to ensure a balanced weight between the string similarity function and the normalized frequency value function where β refers to how much importance we give to the string similarity function with respect to the normalized frequency value function[5].

6. For each $w_{ij} \in R_2$, we calculate the string similarity between w_{ij} and w_i using equation (5), and then assign a weight using the equation (7) only to the words that return the string similarity value greater than 0.5.

7. We sort the words found in step 5 and in step 6 that were given weights, if any, in descending order by the assigned weights and keep only one word as candidate word[6].

3.3.2 Determining Candidate Words for w_2

We use the following steps:

1. We determine the *cut off frequency* for word w_2 as the frequency of the 3-gram $w_1\ w_2\ w_3$ in the Google Web 1T 3-grams, if the said 3-gram exists. Otherwise, we set the *cut off frequency* of w_2 as 0.

2. We find all the 3-grams (where only w_2 is changed while w_1 and w_3 are unchanged) having frequency greater than the *cut off frequency* of w_2 (determined in step 1). Let us consider that we find n replacements of w_2 which are $R_1 = \{w_{21}, w_{22}, \cdots, w_{2n}\}$, and their frequencies $F_1 = \{f_{21}, f_{22}, \cdots, f_{2n}\}$, where f_{2j} is the frequency of the 3-gram $w_1\ w_{2j}\ w_3$.

3. For each $w_{2j} \in R_1$, we calculate the string similarity between w_{2j} and w_2 using equation (5), and then assign a weight using the following equation only to the words that return the string similarity value greater than 0.5.

$$weight = \beta S(w_2, w_{2j}) + (1-\beta)F(w_{2j})$$

4. We sort the words found in step 3 that were given weights, if any, in descending order by the assigned weights and keep only one word as candidate word.

3.3.3 Determining Candidate Words for w_m

We use the following steps:

1. We determine the *cut off frequency* for word w_m as the frequency of the 3-gram $w_{m-2}\ w_{m-1}\ w_m$ in the Google Web 1T 3-grams, if the said 3-gram exists. Otherwise, we set the *cut off frequency* of w_m as 0.

2. We find all the 3-grams (where only w_m is changed while w_{m-2} and w_{m-1} are unchanged) having frequency greater than the *cut off frequency* of w_m (determined in step 1). Let us consider that we find n replacements of w_m which are $R_2 = \{w_{m1}, w_{m2}, \cdots, w_{mn}\}$ and their frequencies $F_2 = \{f_{m1}, f_{m2}, \cdots, f_{mn}\}$, where f_{mj} is the frequency of the 3-gram $w_{m-2}\ w_{m-1}\ w_{mj}$.

3. For each $w_{mj} \in R_2$, we calculate the string similarity between w_{mj} and w_m using equation (5) and then assign a weight using the following equation only to the words that return the string similarity value greater than 0.5.

$$weight = \beta S(w_m, w_{mj}) + (1-\beta)F(w_{mj})$$

4. We sort the words found in step 3 that were given weights, if any, in descending order by the assigned weights and keep only one word as the candidate word.

[5] We give more importance to string similarity function with respect to frequency value function throughout the section of 'determining candidate words' to have more candidate words so that the chance of including the target word into the set of candidate words gets higher. For this reason, we heuristically set β=0.85 in equation (7) instead of setting β=0.5.

[6] Sometimes the top candidate word might be either a plural form or a past participle form of the original word. Or even it might be a high frequency function word (e.g., *the*). We omit these type of words from the candidacy.

4 Evaluation and Experimental Results

We used as test data the same data that Wilcox-O'Hearn et al. (2008) used in their evaluation of Mays et al. (1991) method, which in turn was a replication of the data used by Hirst and St-Onge (1998) and Hirst and Budanitsky (2005) to evaluate their methods.

The data consisted of 500 articles (approximately 300,000 words) from the 1987−89 *Wall Street Journal* corpus, with all headings, identifiers, and so on removed; that is, just a long stream of text. It is assumed that this data contains no errors; that is, the *Wall Street Journal* contains no malapropisms or other typos. In fact, a few typos (both non-word and real-word) were noticed during the evaluation, but they were small in number compared to the size of the text.

Malapropisms were randomly induced into this text at a frequency of approximately one word in 200. Specifically, any word whose base form was listed as a noun in WordNet (but regardless of whether it was used as a noun in the text; there was no syntactic analysis) was potentially replaced by any spelling variation found in the lexicon of the *ispell* spelling checker[7]. A *spelling variation* was defined as any word with an *edit distance* of 1 from the original word; that is, any single-character insertion, deletion, or substitution, or the transposition of two characters, that results in another real word. Thus, none of the induced malapropisms were derived from closed-class words, and none were formed by the insertion or deletion of an apostrophe or by splitting a word. The data contained 1402 inserted malapropisms.

Because it had earlier been used for evaluating Mays et al. (1991)'s trigram method, which operates at the sentence level, the data set had been divided into three parts, without regard for article boundaries or text coherence: sentences into which no malapropism had been induced; the original versions of the sentences that received malapropisms; and the malapropized sentences. In addition, all instances of numbers of various kinds had been replaced by tags such as <INTEGER>, <DOLLAR VALUE>, and <PERCENTAGE VALUE>. Actual (random) numbers or values were restored for these tags. Some spacing anomalies around punctuation marks were corrected. A detailed description of this data can be found in (Hirst, 2008; Wilcox-O'Hearn et al., 2008).

[7]Ispell is a fast screen-oriented spelling checker that shows you your errors in the context of the original file, and suggests possible corrections when it can figure them out. The original was written in PDP-10 assembly in 1971, by R. E. Gorin. The C version was written by Pace Willisson of MIT. Geoff Kuenning added the international support and created the current release.

SUCCESSFUL CORRECTION:
The Iran revelations were particularly disturbing to the Europeans because they came on the heels of the Reykjavik summit between President Reagan and Soviet *reader* → leader [leader] Mikhail Gorbachev.
Even the now sainted Abraham Lincoln was often reviled while in *officer* → office [office], sometimes painted by cartoonists and editorial writers as that baboon in the White House.
FALSE POSITIVE:
· · · by such public displays of interest in *Latinos* → Latin [Latinos], many undocumented · · ·
The southeast Asian nation was one reported *contributor* → contribution [contributor] to the Nicaraguans.
FALSE NEGATIVE:
Kevin Mack, Geldermann president and chief executive officer, didn't return calls for comment on the Clayton *purchaser* [purchase].
U.S. *manufactures* [manufacturers], in short, again are confronting a ball game in which they will be able to play.
TRUE POSITIVE DETECTION, FALSE POSITIVE CORRECTION:
Hawkeye also is known to *rear* → reader [fear] that a bankruptcy-law filing by the parent company, which theoretically shouldn't affect the operations of its member banks, would spark runs on the banks that could drag down the whole entity.
The London Daily News has quoted sources saying as many as 23 British mercenaries were enlisted by KMS to *lid* → slide [aid] the Contras.

Table 2: Examples of successful and unsuccessful corrections. Italics indicate observed word, arrow indicates correction, square brackets indicate intended word.

Some examples of successful and unsuccessful corrections using our proposed method are shown in Table 2.

Table 3 shows our method's results on the described data set compared with the results for the trigram method of Wilcox-O'Hearn et al. (2008)

Detection			correction		
R	P	F_1	R	P	F_1
Lexical cohesion (Hirst and Budanitsky, 2005)					
0.306	0.225	0.260	0.281	0.207	0.238
Trigrams (Wilcox-O'Hearn et al., 2008)					
0.544	0.528	0.536	0.491	0.503	0.497
Multiple 3-grams					
0.890	0.445	0.593	0.763	0.381	0.508

Table 3: A comparison of recall, precision, and F_1 score for three methods of malapropism detection and correction on the same data set.

and the lexical cohesion method of Hirst and Budanitsky (2005). The data shown here for trigram method are not from (Wilcox-O'Hearn et al., 2008), but rather are later results following some corrections reported in (Hirst, 2008). We have not tried optimizing our adjustable parameters: β and αs, because the whole data set was used as testing set by the other methods we compare with. To keep the comparison consistent, we did not use any portion of the data set for training purpose. Having optimized parameters could lead to a better result. The performance is measured using *Recall* (R), *Precision* (P) and F_1:

$$R = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

$$P = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$F_1 = \frac{2PR}{P + R}$$

The fraction of errors correctly detected is the detection *recall* and the fraction of detections that are correct is the detection *precision*. Again, the fraction of errors correctly amended is the correction *recall* and the fraction of amendments that are correct is the correction *precision*. To give an example, consider a sentence from the data set: "The Philippine president, in her commencement address at the academy, complained that the U.S. was *living* → giving [giving] advice instead of the *aid* → said [aid] it pledged.", where italics indicate the observed word, arrow indicates the correction and the square brackets indicate the intended word. The detection *recall* of this sentence is 1.0 and the *precision* is 0.5. The correction *recall* of this sentence is 1.0 and the *precision* is 0.5. For both cases, the F_1 score is 0.667.

We loose some precision because our method tries to detect and correct errors for all the words (except the first word) in the input sentence, and, as a result, it generates more false positives than the other methods. Even so, we get better F_1 scores than the other competing methods. Accepting 8.3 percents extra incorrect detections, we get 34.6 percents extra correct detections of errors, and similarly, accepting 12.2 percents extra incorrect amendments, we get 27.2 percents extra correct amendments of errors compared with the trigrams method (Wilcox-O'Hearn et al., 2008)[8].

5 Conclusion

The Google 3-grams proved to be very useful in detecting real-word errors, and finding the corrections. We did not use the 4-grams and 5-grams because of data sparsity. When we tried with 5-grams the results were lower than the ones presented in Section 4. Having sacrificed a bit the precision score, our proposed method achieves a very good detection recall (0.89) and correction recall (0.76). Our attempts to improve the detection recall or correction recall, while maintaining the respective precisions as high as possible are helpful to the human correctors who post-edit the output of the real-word spell checker. If there is no postediting, at least more errors get corrected automatically. Our method could also detect and correct misspelled words, not only malapropisms, without any modification. In future work, we plan to extend our method to allow for deleted or inserted words, and to find the corrected strings in the Google Web 1T n-grams. In this way we will be able to correct grammar errors too. We also plan more experiments using the 5-grams, but backing off to 4-grams and 3-grams when needed.

Acknowledgments

This work is funded by the Natural Sciences and Engineering Research Council of Canada. We want to thank Professor Graeme Hirst from the Department of Computer Science, University of Toronto, for providing the evaluation data set.

[8]We can run our algorithm on subsets of data to check for variance in the results. We cannot test statistical significance compared to the related work (t-test), because we do not have the system from related work to run it on subsets of the data.

References

L. Allison and T.I. Dix. 1986. A bit-string longest-common-subsequence algorithm. *Information Processing Letters*, 23:305–310.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram corpus version 1.1. Technical report, Google Research.

Lou Burnard, 2000. *Reference Guide for the British National Corpus (World Edition)*, October. www.natcorp.ox.ac.uk/docs/userManual/urg.pdf.

Andrew J. Carlson, Jeffrey Rosen, and Dan Roth. 2001. Scaling up context-sensitive text correction. In *Proceedings of the Thirteenth Conference on Innovative Applications of Artificial Intelligence Conference*, pages 45–50. AAAI Press.

Andrew R. Golding and Dan Roth. 1999. A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

Graeme Hirst and Alexander Budanitsky. 2005. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11(1):87–111, March.

Graeme Hirst and David St-Onge, 1998. *WordNet: An electronic lexical database*, chapter Lexical chains as representations of context for the detection and correction of malapropisms, pages 305–332. The MIT Press, Cambridge, MA.

Graeme Hirst. 2008. An evaluation of the contextual spelling checker of microsoft office word 2007, January. http://ftp.cs.toronto.edu/pub/gh/Hirst-2008-Word.pdf.

Aminul Islam and Diana Inkpen. 2008. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data*, 2(2):1–25.

G. Kondrak. 2005. N-gram similarity and distance. In *Proceedings of the 12h International Conference on String Processing and Information Retrieval*, pages 115–126, Buenos Aires, Argentina.

Karen Kukich. 1992. Technique for automatically correcting words in text. *ACM Comput. Surv.*, 24(4):377–439.

Eric Mays, Fred J. Damerau, and Robert L. Mercer. 1991. Context based spelling correction. *Information Processing and Management*, 27(5):517–522.

I. D. Melamed. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25(1):107–130.

G.A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller. 1993. Introduction to wordnet: An on-line lexical database. Technical Report 43, Cognitive Science Laboratory, Princeton University, Princeton, NJ.

Jennifer Pedler. 2007. *Computer Correction of Real-word Spelling Errors in Dyslexic Text*. Ph.D. thesis, Birkbeck, London University.

Suzan Verberne. 2002. Context-sensitive spell checking based on word trigram probabilities. Master's thesis, University of Nijmegen, February-August.

L. Amber Wilcox-O'Hearn, Graeme Hirst, and Alexander Budanitsky. 2008. Real-word spelling correction with trigrams: A reconsideration of the mays, damerau, and mercer model. In Alexander Gelbukh, editor, *Proceedings, 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2008) (Lecture Notes in Computer Science 4919, Springer-Verlag)*, pages 605–616, Haifa, February.

Semi-supervised Speech Act Recognition in Emails and Forums

Minwoo Jeong[†*] **Chin-Yew Lin**[‡] **Gary Geunbae Lee**[†]
[†]Pohang University of Science & Technology, Pohang, Korea
[‡]Microsoft Research Asia, Beijing, China
[†]{stardust,gblee}@postech.ac.kr [‡]cyl@microsoft.com

Abstract

In this paper, we present a semi-supervised method for automatic speech act recognition in email and forums. The major challenge of this task is due to lack of labeled data in these two genres. Our method leverages labeled data in the Switchboard-DAMSL and the Meeting Recorder Dialog Act database and applies simple domain adaptation techniques over a large amount of unlabeled email and forum data to address this problem. Our method uses automatically extracted features such as phrases and dependency trees, called subtree features, for semi-supervised learning. Empirical results demonstrate that our model is effective in email and forum speech act recognition.

1 Introduction

Email and online forums are important social media. For example, thousands of emails and posts are created daily in online communities, e.g., Usenet newsgroups or the TripAdvisor travel forum[1], in which users interact with each other using emails/posts in complicated ways in discussion threads. To uncover the rich interactions in these email exchanges and forum discussions, we propose to apply speech act recognition to email and forum threads.

Despite extensive studies of speech act recognition in many areas, developing speech act recognition for online forms of conversation is very challenging. A major challenge is that emails and forums usually have no labeled data for training statistical speech act recognizers. Fortunately, labeled speech act data are available in other domains (i.e., telephone and meeting conversations in this paper) and large unlabeled data sets can be collected from the Web. Thus, we focus on the problem of how to accurately recognize speech acts in emails and forums by making maximum use of data from existing resources.

Recently, there are increasing interests in speech act recognition of online text-based conversations. Analysis of speech acts for online chat and instant messages and have been studied in computer-mediated communication (CMC) and distance learning (Twitchell et al., 2004; Nastri et al., 2006; Rosé et al., 2008). In natural language processing, Cohen et al. (2004) and Feng et al. (2006) used speech acts to capture the intentional focus of emails and discussion boards. However, they assume that enough labeled data are available for developing speech act recognition models.

A main contribution of this paper is that we address the problem of learning speech act recognition in a semi-supervised way. To our knowledge, this is the first use of semi-supervised speech act recognition in emails and online forums. To do this, we make use of labeled data from spoken conversations (Jurafsky et al., 1997; Dhillon et al., 2004). A second contribution is that our model learns subtree features that constitute discriminative patterns: for example, variable length n-grams and partial dependency structures. Therefore, our model can capture both local features such as n-grams and non-local dependencies. In this paper, we extend subtree pattern mining to the semi-supervised learning problem.

This paper is structured as follows. Section 2 reviews prior work on speech act recognition and Section 3 presents the problem statement and our data sets. Section 4 describes a supervised method of learning subtree features that shows the effectiveness of subtree features on labeled data sets. Section 5 proposes semi-supervised learning techniques for speech act recognition and Section 6 demonstrates our method applied to email and on-

[*]This work was conducted during the author's internship at Microsoft Research Asia.

[1]http://tripadvisor.com/

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1250–1259,
Singapore, 6-7 August 2009.

line forum thread data. Section 7 concludes this paper with future work.

2 Related Work

Speech act theory is fundamental to many studies in discourse analysis and pragmatics (Austin, 1962; Searle, 1969). A speech act is an illocutionary act of conversation and reflects shallow discourse structures of language. Recent research on spoken dialog processing has investigated computational speech act models of human-human and human-computer conversations (Stolcke et al., 2000) and applications of these models to CMC and distance learning (Twitchell et al., 2004; Nastri et al., 2006; Rosé et al., 2008).

Our work in this paper is closely related to prior work on email and forum speech act recognition. Cohen et al. (2004) proposed the notion of 'email speech act' for classifying the intent of an email sender. They defined verb and noun categories for email speech acts and used supervised learning to recognize them. Feng et al. (2006) presented a method of detecting conversation focus based on the speech acts of messages in discussion boards. Extending Feng et al. (2006)'s work, Ravi and Kim (2007) applied speech act classification to detect unanswered questions. However, none of these studies have focused on the semi-supervised speech act recognition problem and examined their methods across different genres.

The speech processing community frequently employs two large-scale corpora for speech act annotation: Switchboard-DAMSL (SWBD) and Meeting Recorder Dialog Act (MRDA). SWBD is an annotation scheme and collection of labeled dialog act[2] data for telephone conversations (Jurafsky et al., 1997). The main purpose of SWBD is to acquire stochastic discourse grammars for training better language models for automatic speech recognition. More recently, an MRDA corpus has been adapted from SWBD but its tag set for labeling meetings has been modified to better reflect the types of interaction in multi-party face-to-face meetings (Dhillon et al., 2004). These two corpora have been extensively studied, e.g., (Stolcke et al., 2000; Ang et al., 2005; Galley et al., 2004). We also use these for our experiments.

[2]A dialog act is the meaning of an utterance at the level of illocutionary force (Austin, 1962), and broadly covers the speech act and adjacency pair (Stolcke et al., 2000). In this paper, we use only the term 'speech act' for clarity.

This paper focuses on the problem of semi-supervised speech act recognition. The goal of semi-supervised learning techniques is to use auxiliary data to improve a model's capability to recognize speech acts. The approach in Tur et al. (2005) presented semi-supervised learning to employ auxiliary unlabeled data in call classification, and is closely related to our work. However, our approach uses the most discriminative subtree features, which is particularly attractive for reducing the model's size. Our problem setting is closely related to the domain adaptation problem (Ando and Zhang, 2005), i.e., we seek to obtain a model that analyzes target domains (emails and forums) by adapting a method that analyzes source domains (SWBD and MRDA). Recently, this type of domain adaptation has become an important topic in natural language processing.

3 Problem Definition

3.1 Problem Statement

We define speech act recognition to be the task that, given a sentence, maps it to one of the speech act types. Figure 1 shows two examples of our email and forum speech act recognition. E1∼6 are all sentences in an email message. F1∼3, F4∼5, and F6 are three posts in a forum thread. A sentence interacts alone or with others, for example, F6 agrees with the previous post (F4∼5). To gain insight into our work, it is useful to consider that E2, 3 and F1, 4, 6 are summaries of two discourses. In particular, F1 denotes a question and F4 and F6 are corresponding answers. More recently, using speech acts has become an appealing approach in summarizing the discussions (Galley et al., 2004; McKeown et al., 2007).

Next, we define speech act category based on MRDA. Dhillon et al. (2004) included definitions of speech acts for colloquial style interactions (e.g., backchannel, disruption, and floorgrabber), but these are not applicable in emails and forums. After removing these categories, we define 12 tags (Table 1). Dhillon et al. (2004) provides detailed descriptions of each tag. We note that our tag set definition is different from (Cohen et al., 2004; Feng et al., 2006; Ravi and Kim, 2007) for two reasons. First, prior work primarily interested in the domain-specific speech acts, but our work use domain-independent speech act tags. Second, we focus on speech act recognition on the sentence-level.

E1: I am planning my schedule at CHI 2003 (http://www.chi2003.org/)	S
E2: - will there be anything happening at the conference related to this W3C User interest group?	QY
E3: I do not see anything on the program yet, but I suspect we could at least have an informal SIG	S
E4: - a chance to meet others and bring someone like me up to speed on what is happening.	S
E5: There will be many competing activities, so the sooner we can set this up the more likely I can attend.	S
E6: Keith	S
F1: If given a choice, should I choose Huangpu area, or should I choose Pudong area?	QR
F2: Both location are separated by a Huangpu river, not sure which area is more convenient for sight seeing?	QW
F3: Thanks in advance for reply!	P
F4: Stay on the Puxi side of the Huangpu river and visit the Pudong side by the incredible tourist tunnel.	AC
F5: If you stay on the Pudong side add half an hour to visit the majority of the tourist attractions.	S
F6: I definitely agree with previous post.	AA

Figure 1: Examples of speech act recognition in emails and online forums. Tags are defined in Table 1.

Table 1: Tags used to describe components of speech acts

Tag	Description
A	Accept response
AA	Acknowledge and appreciate
AC	Action motivator
P	Polite mechanism
QH	Rhetorical question
QO	Open-ended question
QR	Or/or-clause question
QW	Wh-question
QY	Yes-no question
R	Reject response
S	Statement
U	Uncertain response

The goal of semi-supervised speech act recognition is to learn a classifier using both labeled and unlabeled data. We formally define our problem as follows. Let $\mathbf{x} = \{x_j\}$ be a *forest*, i.e., a set of trees that represents a natural language structure, for example, a sequence of words and a dependency parse tree. We will describe this in more detail in Section 4. Let y be a *speech act*. Then, we define $\mathcal{D}_L = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$ as the set of labeled training data, and $\mathcal{D}_U = \{\mathbf{x}_i\}_{i=n+1}^{l}$ as the set of unlabeled training data where $l = n + m$ and m is the number of unlabeled data instances. Our goal is to find a learning method to minimize the classification errors in $\mathcal{D}_L$ and $\mathcal{D}_U$.

3.2 Data Preparation

In this paper, we separate labeled ($\mathcal{D}_L$) and unlabeled data ($\mathcal{D}_U$). First we use SWBD[3] and MRDA[4] as our labeled data. We automatically map original annotations in SWBD and MRDA to one of the 12 speech acts.[5] Inter-annotator agreement κ in both data sets is ~ 0.8 (Jurafsky et al., 1997; Dhillon et al., 2004). For evaluation purposes, we divide labeled data into three sets: training, development, and evaluation sets (Table 2). Of the 1,155 available conversations in the SWBD corpus, we use 855 for training, 100 for development, and 200 for evaluation. Among the 75 available meetings in the MRDA corpus, we exclude two meetings of different natures (btr001 and btr002). Of the remaining meetings, we use 59 for training, 6 for development, and 8 for evaluation. Then we merge multi-segments utterances that belong to the same speaker and then divide all data sets into sentences.

As stated earlier, our unlabeled data consists of email (EMAIL) and online forum (FORUM) data. For the EMAIL set, we selected 22,391 emails from Enron data[6] (discussion_threads, all_documents, and calendar folders). For the FORUM set, we crawled 11,602 threads and 55,743 posts from the TripAdvisor travel forum site (Beijing, Shanghai, and Hongkong forums). As our evaluation sets, we used 40 email threads of the BC3 corpus[7] for EMAIL and 100 threads selected from the same travel forum site for FORUM. Every sentences was automatically segmented by the MSRA sentence boundary detector (Table 2). Annotation was performed by two human annotators, and inter-annotator agreements were $\kappa = 0.79$ for EMAIL and $\kappa = 0.73$ for FORUM.

Overall performance of automatic evaluation measures usually depends on the distribution of tags. In both labeled and unlabeled sets, the most

[3]LDC Catalog No. LDC97S62

[4]http://www.icsi.berkeley.edu/~ees/dadb/

[5]Our mapping tables are available at http://home.postech.ac.kr/~stardust/acl09/.

[6]http://www.cs.cmu.edu/~enron/

[7]http://www.cs.ubc.ca/nest/lci/bc3.html

Table 2: Number of sentences in labeled and unlabeled data

Set	SWBD	MRDA
Training	96,553	50,865
Development	12,299	8,366
Evaluation	24,264	10,492

Set	EMAIL	FORUM
Unlabeled	122,125	297,017
Evaluation	2,267	3,711

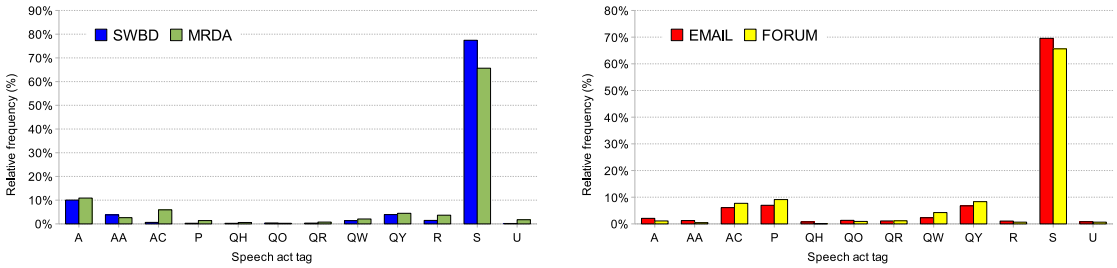


Figure 2: Distribution of speech acts in the evaluation sets. Tags are defined in Table 1.

frequent tag is the statement (S) tag (Figure 2). Distributions of tags are similar in training and development sets of SWBD and MRDA.

4 Speech Act Recognition

Previous work in speech act recognition used a large set of lexical features, e.g., bag-of-words, bigrams and trigrams (Stolcke et al., 2000; Cohen et al., 2004; Ang et al., 2005; Ravi and Kim, 2007). However, these methods create a large number of lexical features that might not be necessary for speech act identification. For example, a Wh-question "What site should we use to book a Beijing-Chonqing flight?" can be predicted by two discriminative features, "(<s>, WRB) → QW" and "(?, </s>) → QW" where <s> and </s> are sentence start and end symbols, and WRB is a part-of-speech tag that denotes a Wh-adverb. In addition, useful features could be of various lengths, i.e. not fixed length n-grams, and non-adjacent. One key idea of this paper is a novel use of subtree features to model these for speech act recognition.

4.1 Exploiting Subtree Features

To exploit subtree features in our model, we use a *subtree pattern mining* method proposed by Kudo and Matsumoto (2004). We briefly introduce this algorithm here. In Section 3.1, we defined $\mathbf{x} = \{x_j\}$ as the forest that is a set of trees. More precisely, x_j is a labeled ordered tree where each node has its own label and is ordered left-to-right. Several types of labeled ordered trees

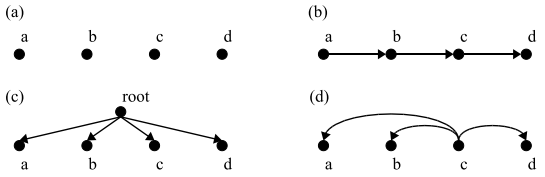


Figure 3: Representations of tree: (a) bag-of-words, (b) n-gram, (c) word pair, and (d) dependency tree. A node denotes a word and a directed edge indicates a parent-and-child relationship.

are possible (Figure 3). Note that S-expression can be used instead for computation, for example `(a(b(c(d))))` for the n-gram (Figure 3(b)). Moreover, we employ a combination of multiple trees as the input of the subtree pattern mining algorithm.

We extract subtree features from the forest set $\{\mathbf{x}_i\}$. A subtree t is a tree if $t \subseteq \mathbf{x}$. For example, `(a)`, `(a(b))`, and `(b(c(d)))` are subtrees of Figure 3(b). We define the subtree feature as a weak learner:

$$f(y, t, \mathbf{x}) \triangleq \begin{cases} +y & t \subseteq \mathbf{x}, \\ -y & \text{otherwise}, \end{cases} \quad (1)$$

where we assume a binary case $y \in \mathcal{Y} = \{+1, -1\}$ for simplicity. Even though the approach in Kudo and Matsumoto (2004) and ours are simiar, there are two clear distinctions. First, our method employs multiple tree structures, and uses different constraints to generate subtree candidates. In this paper, we only restrict generating

the dependency subtrees which should have 3 or more nodes. Second, our method is of interest for semi-supervised learning problems. To learn subtree features, Kudo and Matsumoto (2004) assumed supervised data $\{(\mathbf{x}_i, y_i)\}$. Here, we describe the supervised learning method and will describe our semi-supervised method in Section 5.

4.2 Supervised Boosting Learning

Given training examples, we construct a ensemble learner $F(\mathbf{x}) = \sum_k \lambda_k f(y_k, t_k, \mathbf{x})$, where λ_k is a coefficient for linear combination. A final classifier $h(\mathbf{x})$ can be derived from the ensemble learner, i.e., $h(\mathbf{x}) \triangleq sgn(F(\mathbf{x}))$. As an optimization framework (Mason et al., 2000), the objective of boosting learning is to find F such that the cost of functional

$$\mathcal{C}(F) = \sum_{i \in \mathcal{D}} \alpha_i C[y_i F(\mathbf{x}_i)] \tag{2}$$

is minimized for some non-negative and monotonically decreasing cost function $C : \mathbb{R} \to \mathbb{R}$ and the weight $\alpha_i \in \mathbb{R}^+$. In this paper, we use the AdaBoost algorithm (Schapire and Singer, 1999); thus the cost function is defined as $C(z) = e^{-z}$.

Constructing an ensemble learner requires that the user choose a base learner, $f(y, t, \mathbf{x})$, to maximize the inner product $-\langle \nabla \mathcal{C}(F), f \rangle$ (Mason et al., 2000). Finding $f(y, t, \mathbf{x})$ to maximize $-\langle \nabla \mathcal{C}(F), f \rangle$ is equivalent to searching for $f(y, t, \mathbf{x})$ to minimize $2 \sum_{i: f(y,t,\mathbf{x}_i) \neq y_i} w_i - 1$, where w_i for $i \in \mathcal{D}_L$, is the empirical data distribution $w_i^{(k)}$ at step k. It is defined as:

$$w_i^{(k)} = \alpha_i \cdot e^{-y_i F(\mathbf{x}_i)}. \tag{3}$$

From Eq. 3, a proper base learner (i.e., subtree) can be found by maximizing weighted gain, where

$$\text{gain}(t, y) = \sum_{i \in \mathcal{D}_L} y_i w_i f(y, t, \mathbf{x}_i). \tag{4}$$

Thus, subtree mining is formulated as the problem of finding $(\hat{t}, \hat{y}) = \underset{(t,y) \in \mathcal{X} \times \mathcal{Y}}{\arg\max}\ \text{gain}(t, y)$. We need to search with respect to a non-monotonic score function (Eq. 4), thus we use the monotonic bound, $\text{gain}(t, y) \leq \mu(t)$, where

$$\mu(t) = \max \left(2 \sum_{\{i | y_i = +1, t \subseteq \mathbf{x}_i\}} w_i - \sum_{i=1}^{n} y_i f(y, t, \mathbf{x}_i), \right.$$
$$\left. 2 \sum_{\{i | y_i = -1, t \subseteq \mathbf{x}_i\}} w_i + \sum_{i=1}^{n} y_i f(y, t, \mathbf{x}_i) \right). \tag{5}$$

Table 3: Result of supervised learning experiment; columns are micro-averaged F_1 score with macro-averaged F_1 score in parentheses. MAXENT: maximum entropy model; BOW: bag-of-words model; NGRAM: n-gram model; +POSTAG, +DEPTREE, +SPEAKER indicate that the components were added individually onto NGRAM. '*' indicates results significantly better than the NGRAM model ($p < 0.001$).

Model	SWBD	MRDA
MAXENT	92.76 (63.54)	82.48 (57.19)
BOW	91.32 (54.47)	82.17 (55.42)
NGRAM	92.60 (58.43)	83.30 (57.53)
+POSTAG	92.69 (60.07)	83.60 (58.46)
+DEPTREE	92.67 (61.75)	*83.57 (57.45)
+SPEAKER	*92.86 (63.13)	83.40 (58.20)
ALL	*92.87 (63.77)	83.49 (59.04)

The subtree set is efficiently enumerated using a branch-and-bound procedure based on $\mu(t)$ (Kudo and Matsumoto, 2004).

After finding an optimal base leaner, $f(\hat{y}, \hat{t}, \mathbf{x})$, we need to set the coefficient λ_k to form a new ensemble, $F(\mathbf{x}_i) \leftarrow F(\mathbf{x}_i) + \lambda_k f(\hat{t}, \hat{y}, \mathbf{x}_i)$. In AdaBoost, we choose

$$\lambda_k = \frac{1}{2} \log \left(\frac{1 + \text{gain}(\hat{t}, \hat{y})}{1 - \text{gain}(\hat{t}, \hat{y})} \right). \tag{6}$$

After K iterations, the boosting algorithm returns the ensemble learner $F(\mathbf{x})$ which consists of a set of appropriate base learners $f(y, t, \mathbf{x})$.

4.3 Evaluation on Labeled Data

We verified the effectiveness of using subtree features on the SWBD and MRDA data sets. For boosting learning, one typically assumes $\alpha_i = 1$. In addition, the number of iterations, which relates to the number of patterns, was determined by a development set. We also used a one-vs.-all strategy for the multi-class problem. Precision and recall were computed and combined into micro- and macro-averaged F_1 scores. The significance of our results was evaluated using the McNemar paired test (Gillick and Cox, 1989), which is based on individual labeling decisions to compare the correctness of two models. All experiments were implemented in C++ and executed in Windows XP on a PC with a Dual 2.1 GHz Intel Core2 processor and 2.0 Gbyte of main memory.

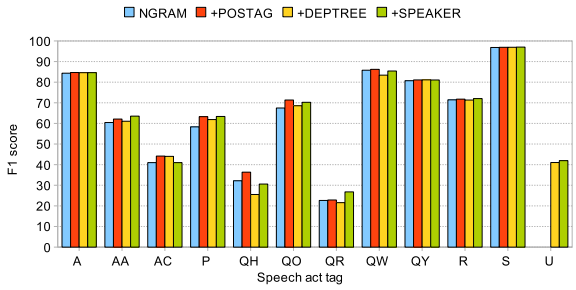


Figure 4: Comparison of different trees (SWBD)

We show that use of subtree features is effective to solve the supervised speech act recognition problem. We also compared our model with the state-of-the-art maximum entropy classifier (MAXENT). We used bag-of-words, bigram and trigram features for MAXENT, which modeled 702k (SWBD) and 460k (MRDA) parameters (i.e., patterns), and produced micro-averaged F_1 scores of 92.76 (macro-averaged F_1 = 63.54) for SWBD and 82.48 (macro-averaged F_1 = 57.19) for MRDA. In contrast, our method generated approximately 4k to 5k patterns on average with similar or greater F_1 scores (Table 3); hence, compared to MAXENT, our model requires fewer calculations and is just as accurate.

The n-gram model (NGRAM) performed significantly better than the bag-of-words model (McNemar test; $p < 0.001$) (Table 3). Unlike MAXENT, NGRAM automatically selects a relevant set of variable length n-gram features (i.e., phrase features). To this set, we separately added two syntax type features, part-of-speech tag n-gram (POSTAG) and dependency parse tree (DEPTREE) automatically parsed by Minipar[8], and one discourse type feature, speaker n-gram (SPEAKER). Although some micro-averaged F_1 are not statistically significant between the original NGRAM and the models that include POSTAG, DEPTREE or SPEAKER, macro-averaged F_1 values indicate that minor classes can take advantage of other structures. For example, in the result of SWBD (Figure 4), DEPTREE and SPEAKER models help to predict uncertain responses (U), whereas NGRAM and POSTAG cannot do this.

5 Semi-supervised Learning

Our goal is to eventually make maximum use of existing resources in SWBD and MRDA for email/forum speech act recognition. We call the model trained on the mixed data of these two corpora BASELINE. We use ALL features in constructing the BASELINE for the semi-supervised experiments. While this model gave promising results using SWBD and MRDA, language used in emails and forums differs from that used in spoken conversation. For example, 'thanx' is an expression commonly used as a polite mechanism in online communications. To adapt our model to understand this type of difference between spoken and online text-based conversations, we should induce new patterns from unlabeled email and forum data. We describe here two methods of semi-supervised learning.

5.1 Method 1: Bootstrapping

First, we bootstrap the BASELINE model using automatically predicted unlabeled examples. However, using all of the unlabeled data results in noisy models; therefore filtering or selecting data is very important in practice. To this end, we only select similar examples by criterion, $d(\mathbf{x}_i, \mathbf{x}_j) < r$ or k nearest neighbors where $\mathbf{x}_i \in \mathcal{D}_L$ and $\mathbf{x}_j \in \mathcal{D}_U$. In practice, r or k are fixed. In our method, examples are represented by trees; hence we use a "tree edit distance" for calculating $d(\mathbf{x}_i, \mathbf{x}_j)$ (Shasha and Zhang, 1990). Selected examples are evaluated using BASELINE, and using subtree pattern mining runs on the augmented data (i.e. unlabeled). We call this method BOOTSTRAP.

5.2 Method 2: Semi-supervised Boosting

Our second method is based on a principle of semi-supervised boosting learning (Bennett et al., 2002). Because we have no supervised guidance for $\mathcal{D}_U$, our objective functional to find F is defined as:

$$\mathcal{C}(F) = \sum_{i \in \mathcal{D}_L} \alpha_i C[y_i F(\mathbf{x}_i)] + \sum_{i \in \mathcal{D}_U} \beta_i C[|F(\mathbf{x}_i)|] \quad (7)$$

This cost functional is non-differentiable. To solve it, we introduce pseudo-labels $\tilde{y}$ where $\tilde{y} = sgn(F(\mathbf{x}))$ and $|F(\mathbf{x})| = \tilde{y}F(\mathbf{x})$. Using the same derivation in Section 4.2, we obtain the following

[8] http://www.cs.ualberta.ca/∼lindek/minipar.htm

gain function and update rules:

$$\text{gain}(t, y) = \sum_{i \in \mathcal{D}_L} y_i w_i f(y, t, \mathbf{x}_i) + \sum_{i \in \mathcal{D}_U} \tilde{y}_i w_i f(y, t, \mathbf{x}_i), \quad (8)$$

$$w_i = \begin{cases} \alpha_i \cdot e^{-y_i F(\mathbf{x}_i)} & i \in \mathcal{D}_L, \\ \beta_i \cdot e^{-\tilde{y}_i F(\mathbf{x}_i)} & i \in \mathcal{D}_U. \end{cases} \quad (9)$$

Intuitively, an unlabeled example that has a high-confidence $|F(\mathbf{x})|$ at the current step, will probably receive more weight at the next step. That is, similar instances become more important when learning and mining subtrees. This semi-supervised boosting learning iteratively generates pseudo-labels for unlabeled data and finds the value of F that minimizes training errors (Bennett et al., 2002). Also, the algorithm infers new features from unlabeled data, and these features are iteratively re-evaluated by the current ensemble learner. We call this method SEMIBOOST.

6 Experiment

6.1 Setting

We describe specific settings used in our experiment. Because we have no development set, we set the maximum number of iterations K at 10,000. At most K patterns can be extracted, but this seldom happens because duplicated patterns are merged. Typical settings for semi-supervised boosting are $\alpha_i = 1$ and $\beta_i = 0.5$, that is, we penalize the weights for unlabeled data.

For efficiency, BASELINE model used 10% of the SWBD and MRDA data, selected at random. We observed that this data set does not degrade the results of semi-supervised speech act recognition. For BOOTSTRAP and SEMIBOOST, we selected $k = 100$ nearest neighbors of unlabeled examples for each labeled example using tree edit distance, and then used 24,625 (SWBD) and 54,961 (MRDA) sentences for the semi-supervised setting.

All trees were combined as described in Section 4.3 (ALL model). In EMAIL and FORUM data we added different types of discourse features: message type (e.g., initial or reply posts), authorship (e.g., an identification of 2nd or 3rd posts written by the same author), and relative position of a sentence. In Figure 1, for example, F1∼3 is an initial post, and F4∼5 and F6 are reply posts. Moreover, F1, F4, and F6 are the first sentence in each post.

Table 4: Results of speech act recognition on online conversations; columns are micro-averaged F_1 score with macro-averaged scores in parentheses. '⋆' indicates that the result is significantly better than BASELINE ($p < 0.001$).

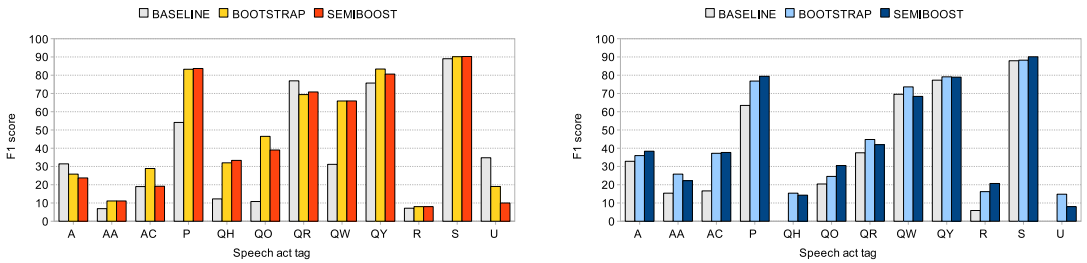
Model	EMAIL	FORUM
BASELINE	78.87 (37.44)	78.93 (35.57)
BOOTSTRAP	⋆83.11 (44.90)	79.09 (44.38)
SEMIBOOST	⋆82.80 (44.64)	⋆81.76 (44.21)
SUPERVISED	90.95 (75.71)	83.67 (40.68)

These features do not occur in SWBD or MRDA because these are utterance-by-utterance conversations.

6.2 Result and Discussion

First, we show that our method of semi-supervised learning can improve modeling of the speech act of emails and forums. As our baseline, BASELINE achieved a micro-averaged F_1 score of $\sim$ 79 for both data sets. This implies that SWBD and MRDA data are useful for our problem. Using unlabeled data, semi-supervised methods BOOTSTRAP and SEMIBOOST perform better than BASELINE (Table 4; Figure 5). To verify our claim, we evaluated the supervised speech act recognition on EMAIL and FORUM evaluation sets with 5-fold cross validation (SUPERVISED in Table 4). In particular, our semi-supervised speech act recognition is competitive with the supervised model in FORUM data.

The difference in performance between supervised results in EMAIL and FORUM seems to indicate that the latter is a more difficult data set. However, our SEMIBOOST method were able to come close to the supervised FORUM results (81.76 vs. 83.67). This is also close to the range of supervised MRDA data set ($F_1 = 83.49$ for ALL, Table 3). Moreover, we analyzed a main reason of why transfer results were competitive in the FORUM but not in the EMAIL. This might be due to the mismatch in the unlabeled data, that is, we used different email collections, the BC3 corpus (email communication of W3C on w3.org sites), for evaluation while used Enron data for adaption. We also conjecture that the discrepancy between EMAIL and FORUM is probably due to the more heterogeneous nature of the FORUM data where anyone can post and reply while EMAIL (Enron or



(a) EMAIL (b) FORUM

Figure 5: Result of the semi-supervised learning method

BC3) might have a more fix set of participants.

The improvement of less frequent tags is prominent, for example 25% for action motivator (AC), 40% for polite mechanism (P), and 15% for rhetorical question (QR) error rate reductions were achieved in FORUM data (Figure 5(b)). Therefore, the semi-supervised learning method is more effective with small amounts of labeled data (i.e., less frequent annotations). We believe that despite their relative rarity, these speech acts are more important than the statement (S) in some applications, e.g., summarization.

Next, we give a qualitative analysis for better interpretation of our problem and results. Due to limited space, we focus on FORUM data, which can potentially be applied to many applications. Of the top ranked patterns extracted by SEMIBOOST (Figure 6(a)), subtree patterns of n-gram, part-of-speech, dependency parse trees are most discriminative. The patterns from unlabeled data have relatively lower ranks, but this is not surprising. This indicates that BASELINE model provides the base knowledge for semi-supervised speech act recognition. Also, unlabeled data for EMAIL and FORUM help to induce new patterns or adjust the model's parameters. As a result, the semi-supervised method is better than the BASELINE when an identical number of patterns is modeled (Figure 6(b)). For this result, we conclude that our method successfully transfers knowledge from a source domain (i.e., SWBD and MRDA) to a target domain (i.e., EMAIL and FORUM); hence it can be a solution to the domain adaption problem.

Finally, we determine the main reasons for error (in SEMIBOOST), to gain insights that may allow development of better models in future work (Figure 6(c)). We sorted speech act tags by their semantics and partitioned the confusion matrix into question type (Q*) and statement, which are two high-level speech acts. Most errors occur in the similar categories, that is, language usage in question discourse is definitely distinct from that in statement discourse. From this analysis, we believe that more advanced techniques (e.g. two-stage classification and learning with hierarchy-augmented loss) can improve our model.

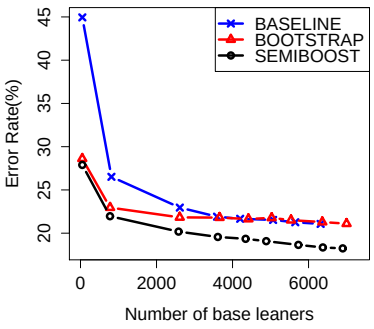
7 Conclusion

Despite the increasing interest in online text-based conversations, no study to date has investigated semi-supervised speech act recognition in email and forum threads. This paper has addressed the problem of learning to recognize speech acts using labeled and unlabeled data. We have also contributed to the development of a novel application of boosting subtree mining. Empirical results have demonstrated that semi-supervised learning of speech act recognition with subtree features improves the performance in email and forum data sets. An attractive future direction is to exploit prior knowledge for semi-supervised speech act recognition. Druck et al. (2008) described generalized expectation criteria in which a discriminative model can employ the labeled features and unlabeled instances. Using prior knowledge, we expect that our model will effectively learn useful patterns from unlabeled data.

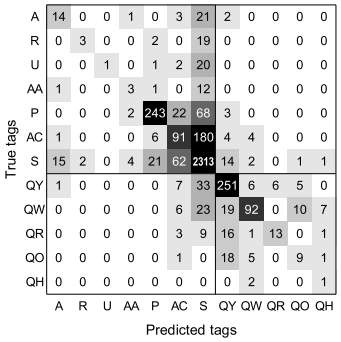
As work progresses on analyzing online text-based conversations such as emails, forums, and online chats, the importance of developing models for discourse without annotating much new data will become more important. In the future, we plan to explore other related problems such as adjacency pairs (Levinson, 1983) and discourse parsing (Soricut and Marcu, 2003) for large-scale online forum data.

Tag	Example pattern
A	(ROOT (yep))
AA	(<s> (wow (. (</s>))))
AC	(WRB (VB (NN (PRP))))
P	(thanks)
QH	(cares (?))
QO	(ROOT (think) (?))
QR	(ROOT (or) (?))
QW	(ROOT (rel=sub (what)))
QY	(<s> (do))
R	(nay)
S	(ROOT (U (,)) (?))
U	(it (is (possible (.))))

(a) Example patterns



(b) Learning behavior



(c) Confusion matrix

Figure 6: Analysis on FORUM data

Acknowledgement

We would like to thank to anonymous reviewers for their valuable comments, and Yunbo Cao, Wei Lai, Xinying Song, Jingtian Jing, and Wei Wu for their help in preparing our data.

References

R. Ando and T. Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

J. Ang, Y. Liu, and E. Shriberg. 2005. Automatic dialog act segmentation and classification in multiparty meetings. In *Proceedings of ICASSP*, pages 1061–106.

J. Austin. 1962. *How to Do Things With Words*. Harvard Univ. Press, Cambridge, MA.

K.P. Bennett, A. Demiriz, and R. Maclin. 2002. Exploiting unlabeled data in ensemble methods. In *Proceedings of ACM SIGKDD*, pages 289–296.

W.W. Cohen, V.R. Carvalho, and T. Mitchell. 2004. Learning to classify email into "speech acts". In *Proceedings of EMNLP*, pages 309–316.

R. Dhillon, S. Bhagat, H. Carvey, and E. Shriberg. 2004. Meeting recorder project: Dialog act labeling guide. Technical report, International Computer Science Institute.

G. Druck, G. Mann, and A. McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of ACM SIGIR*, pages 595–602.

D. Feng, E. Shaw, J. Kim, and E. H. Hovy. 2006. Learning to detect conversation focus of threaded discussions. In *Proceedings of HLT-NAACL*, pages 208–215.

M. Galley, K. McKeown, J. Hirschberg, and E. Shriberg. 2004. Identifying agreement and disagreement in conversational speech: use of bayesian networks to model pragmatic dependencies. In *Proceedings of ACL*.

L. Gillick and S. Cox. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings of ICASSP*, pages 532–535.

D. Jurafsky, E. Shriberg, and D. Biasca. 1997. Switchboard SWBD-DAMSL labeling project coder's manual, draft 13. Technical report, Univ. of Colorado Institute of Cognitive Science.

T. Kudo and Y. Matsumoto. 2004. A boosting algorithm for classification of semi-structured text. In *Proceedings of EMNLP*, pages 301–308.

S. Levinson. 1983. *Pragmatics*. Cambridge Univ. Press, Cambridge.

L. Mason, P. Bartlett, J. Baxter, and M. Frean. 2000. Functional gradient techniques for combining hypotheses. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 221–246. MIT Press, Cambridge, MA.

K. McKeown, L. Shrestha, and O. Rambow. 2007. Using question-answer pairs in extractive summarization of email conversations. In *Proceedings of CICLing*, volume 4394 of *Lecture Notes in Computer Science*, pages 542–550.

J. Nastri, J. Pe na, and J. T. Hancock. 2006. The construction of away messages: A speech act analysis. *Journal of Computer-Mediated Communication*, 11(4):article 7.

S. Ravi and J. Kim. 2007. Profiling student interactions in threaded discussions with speech act classifiers. In *Proceedings of the AI in Education Conference*.

C. Rosé, Y. Wang, Y. Cui, J. Arguello, K. Stegmann, A. Weinberger, and F. Fischer. 2008. Analyzing collaborative learning processes automatically: Exploiting the advances of computational linguistics in computer-supported collaborative learning. *International Journal of Computer-Supported Collaborative Learning*, 3(3):237–271.

R.E. Schapire and Y. Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

J. Searle. 1969. *Speech Acts*. Cambridge Univ. Press, Cambridge.

D. Shasha and K. Zhang. 1990. Fast algorithms for the unit cost editing distance between trees. *Journal of Algorithms*, 11(4):581–621.

R. Soricut and D. Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of NAACL-HLT*, pages 149–156.

A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. Van Ess-Dykema, and M. Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373.

G. Tur, D. Hakkani-Tür, and R. E. Schapire. 2005. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45(2):171–186.

D. P. Twitchell, J. F. Nunamaker, and J. K. Burgoon. 2004. Using speech act profiling for deception detection. In *Second Symposium on Intelligence and Security Informatics*, volume 3073 of *Lecture Notes in Computer Science*, pages 403–410.

Using Morphological and Syntactic Structures for Chinese Opinion Analysis

Lun-Wei Ku **Ting-Hao Huang** **Hsin-Hsi Chen**

Department of Computer Science and Information Engineering
National Taiwan University
No. 1, Sec. 4, Roosevelt Road, Taipei, 10617 Taiwan
{lwku,thhuang}@nlg.csie.ntu.edu.tw;hhchen@ntu.edu.tw

Abstract

This paper employs morphological structures and relations between sentence segments for opinion analysis on words and sentences. Chinese words are classified into eight morphological types by two proposed classifiers, CRF classifier and SVM classifier. Experiments show that the injection of morphological information improves the performance of the word polarity detection. To utilize syntactic structures, we annotate structural trios to represent relations between sentence segments. Experiments show that considering structural trios is useful for sentence opinion analysis. The best f-score achieves 0.77 for opinion word extraction, 0.62 for opinion word polarity detection, 0.80 for opinion sentence extraction, and 0.54 for opinion sentence polarity detection.

1 Introduction

Sentiment analysis has attracted much attention in recent years because a large scale of subjective information is disseminated through various platforms on the web. Sentiment information can be applied to a wide variety of fields, including product recommendation, review summarization, public polling, and so on.

Opinion dictionaries are important resources for identifying subjective information. Several approaches were proposed to collect such resources. Wiebe (2000) learned subjective adjectives from corpora. Takamura *et al.* (2005) extracted semantic orientations of words. Ku *et al.* (2007) measured sentiment degrees of Chinese words by averaging the sentiment scores of the composing characters. When the opinion words are available, the polarities of sentences and documents can be determined by them. Riloff and Wiebe (2003) learned the extraction patterns for subjective expressions. Kim and Hovy (2004) found the polarity of subjective expressions. Pang *et al.* (2002) and Dave *et al.* (2003) explored various techniques at document level.

Morphological information has been widely used in classifying words, telling the meanings, and doing other in-depth analysis (Tzeng and Chen, 2002). However, morphological information was seldom applied either in Chinese opinion extraction, or in solving the coverage problem of opinion dictionary. Instead of bag-of-characters approach (Ku *et al.*, 2007), this paper employs morphological structures of words to extract opinion words.

Relations between sentence segments are also defined by linguistics in the Chinese language. These are similar to morphological structures between Chinese characters. Based on parsing trees of sentences, we identify these relations and utilize them for opinion analysis on sentences.

As the experimental corpus, some researchers managed to generate annotated materials and gold standards under many constraints. Ku set a standard for generating final answers from annotations of multiple annotators (Ku *et al.*, 2007), and Somasundaran annotated discourse information from meeting dialogs to train a sentiment model (Somasundaran *et al.*, 2007). For multilingual issues, researchers concerned mainly about the applicability of corpus and algorithms from the native language to foreign languages (Banea *et al.*, 2008; Bautin *et al.*, 2008).

Several opinion analysis systems have been developed so far. OASYS (Cesarano *et al.*, 2007) and CopeOpi (Ku *et al.*, 2007) allow users input their queries and select preferred data sources,

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1260–1269,
Singapore, 6-7 August 2009.

and then track opinions in a time zone. For both systems, extracting opinions is the main focus, while holders and targets are identified implicitly when retrieving relevant documents. Carenini's team proposed a graphical user interface for evaluative texts (2006), in which color blocks were used to present the evaluations for components of products. Fair News Reader, a Japanese news Web system, incorporates sentiment information insensibly in an interesting way (Kawai *et al.*, 2007). It provides readers "balanced" reports by analyzing the sentiment in news articles which readers have read, and suggests them new articles according to the analysis results. It leads the application of opinion analysis to the direction of personalization.

2 Chinese Morphological Structures

In the Chinese language, a word is composed of one or more Chinese characters, and its meaning can be interpreted in terms of its composite characters. The morphological structures of Chinese words are formulated by three major processes in linguistics: *compounding*, *affixation*, and *conversion*. *Compounding* is a complex word-formation process. In most cases, two or more morphemes together are formed as a lexical item by this process. *Affixation* is a morphological process, by which grammatical or lexical information is added to a base form. By the *conversion* process, a word is changed from one part of speech into another without the addition or deletion of any morphemes.

Compounding is the most productive way to construct a Chinese word. Mostly, a Chinese character itself carries meanings, so that a morpheme can function as a character and has its own part of speech. In some cases, a Chinese morpheme may carry no specific meaning and just makes a word more readable. Cheng and Tian (1992) divided Chinese words into five morphological types based on the relations between the morphemes in *compounding* words.

(1) Parallel Type: Two morphemes play coordinate roles in a word. For example, the morphemes "財" (money) and "富" (wealth) are parallel in the word "財富" (money-wealth).

(2) Substantive-Modifier Type: A modified morpheme follows a modifying morpheme. For example, the morpheme "哭" (cry) is modified by "痛" (bitterly) in the word "痛哭" (bitterly-cry).

(3) Subjective-Predicate Type: One morpheme is an expresser and the other one is described. The structure is like a subject-verb sentence condensed in one word. For example, the morpheme "心" (heart) is a subject of the predicate "疼" (hurt) in the word "心疼" (heart-hurt).

(4) Verb-Object Type: The first morpheme is usually a verb which governs the second one, making this word similar to a verb followed by its object. For example, the morpheme "控" (control) serves as the object of the verb "失" (lose) in the word "失控" (lose-control).

(5) Verb-Complement Type: The first morpheme is usually a verb but sometimes can be an adjective, and the second morpheme explains the first one from different aspects. For example, the morpheme "清" (clearly) expresses the aspects of the action "看" (look).

Chinese words constructed by *affixation* process can be one of the two cases – say, *morpheme* and *morpheme*, or *morpheme* and *non-morpheme*. In the case of *morpheme* and *morpheme*, the affixation word belongs to one of the above 5 types if the prefix and the suffix are neither negations nor confirmations. Types 6 and 7 defined below represent the affixation words whose prefix or suffix is a negation or a confirmation. The affixation words whose prefix or suffix characters are not morphemes are classified into type 8.

(6) Negation Type: There is at least one negation character in words of this type. For example, the prefix "無" (no) is the negation morpheme in the word "無法" (no-method).

(7) Confirmation Type: There is at least one confirmation character in words of this type. For example, the prefix "有" (do) is a confirmation in the word "有賴" (do-depend on).

(8) Others: Those words that do not belong to the above seven types are assigned to this type, such as words whose meanings are not a function of their composite characters, words whose composite characters are not morphemes, such as "姪子" (nephew-suffix) and "薄荷" (peppermint).

3 Opinion Scores of Chinese Words

The bag-of-characters approach proposed by Ku *et al*. (2007) considers the observation probabilities of characters in Chinese opinion words. It calculates the observation probabilities of characters from a set of seeds first, then dynamically enlarges the set and adjusts their probabilities. In

this approach, the opinion score of a word is determined by the combination of the observation probabilities of its composite characters defined by Formulas (1) and (2).

$$P(C, pos) = \frac{f(C, pos) / \sum_{i=1}^{n} f(C_i, pos)}{f(C, pos) / \sum_{i=1}^{n} f(C_i, pos) + f(C, neg) / \sum_{i=1}^{m} f(C_i, neg)} \quad (1)$$

$$P(C, neg) = \frac{f(C, neg) / \sum_{i=1}^{m} f(C_i, neg)}{f(C, pos) / \sum_{i=1}^{n} f(C_i, pos) + f(C, neg) / \sum_{i=1}^{m} f(C_i, neg)} \quad (2)$$

$$S(C) = P(C, pos) - N(C, neg) \quad (3)$$

$$S(C_1C_2...C_l) = \frac{1}{l}\sum_{i=1}^{l} S(C_i) \quad (4)$$

where C is an arbitrary Chinese character, $f(C, polarity)$ counts the observed frequency of C in a set of Chinese words whose opinion polarity is positive (*pos*) or negative (*neg*); $P(C, pos)$ and $P(C, neg)$ denote the observation probabilities of C as a positive and a negative character, and n and m denote total number of unique characters in positive and negative words. The difference of $P(C, pos)$ and $P(C, neg)$ in Formula (3) determines the sentiment score of character C, denoted by $S(C)$. Formula (4) computes the opinion score of a word of l characters $C_1C_2…C_l$ by averaging their scores.

Instead of counting the weights as in the bag-of-characters approaches, we consider the word structures and propose a scoring function for each morphological type. According to the Frequency Dictionary of Modern Chinese, 96.5% of Chinese words are unigrams and bigrams (Chen, *et al.*, 1997). In the following functions, $S(C_1C_2)$ computes the opinion scores of words with characters C_1 and C_2. $SIGN(s)$ returns -1 if polarity degree s is smaller than 0, i.e., negative, and returns 1 when positive.

(1) Parallel Type: Since the two composite characters of a word of this type are homogeneous, the opinion score is the average score of two characters' opinion scores.

$$S(C_1C_2) = \frac{S(C_1) + S(C_2)}{2} \quad (5)$$

(2) Substantive-Modifier Type: The first morpheme of a word of this type modifies the second one, so that its opinion weight comes from the absolute opinion score of the first character, while the opinion polarity is determined by the occurrence of negative opinion characters. If at least one negative opinion character appears, the word is negative, else it is positive. For example, the word "痛哭" (bitterly cry) is composed of "痛" (bitterly, negative) and "哭" (cry, negative). Negative characters make this word negative and its opinion strength, i.e., the absolute value of the score, is decided by the first character for the degree of crying.

$$\begin{aligned} &\text{if } (S(C_1) \neq 0 \text{ and } S(C_2) \neq 0) \text{ then} \\ &\quad \text{if } (S(C_1) > 0 \text{ and } S(C_2) > 0) \text{ then } S(C_1C_2) = S(C_1) \\ &\quad \text{else } S(C_1C_2) = -1 \times |S(C_1)| \\ &\text{else } S(C_1C_2) = S(C_1) + S(C_2) \end{aligned} \quad (6)$$

(3) Subjective-Predicate Type: The first morpheme of a word of this type is a subject and the second morpheme is the action it performs, so that the action decides the opinion score of the word. If the action is not an opinion or it is neutral, the subject determines the opinion score of this word. For example, the word "山崩" (mudslide, negative) is composed of "山" (mountain, non-opinion) and "崩" (collapse, negative). Its opinion score depends only on the second character "崩" (collapse) since the first character is a subject and usually bears no opinions.

$$\begin{aligned} &\text{if } (S(C_2) \neq 0) \text{ then } S(C_1C_2) = S(C_2) \\ &\text{else } S(C_1C_2) = S(C_1) \end{aligned} \quad (7)$$

(4) Verb-Object Type: The first morpheme of words of this type acts upon the second morpheme. The effect depends not only on the action but on the target. The weight is determined by the action, but the polarity is the multiplication of the signs of the two morphemes. For example, the word "避暑" (to go away for the summer, positive) is composed of "避" (hide, negative) and "暑" (hot summer, negative). Its strength depends on the strength of "避" (hide) and polarity is positive from the multiplication of two negatives.

$$\begin{aligned} &\text{if } (S(C_1) \neq 0 \text{ and } S(C_2) \neq 0) \\ &\quad \text{then } S(C_1C_2) = |S(C_1)| \times SIGN(S(C_1)) \times SIGN(S(C_2)) \\ &\quad \text{else } S(C_1C_2) = S(C_1) + S(C_2) \end{aligned} \quad (8)$$

(5) Verb-Complement Type: The scoring function for words of this type is defined the same as that of a Subjective-Predicate type in Formula (7). The complement morpheme is the deciding factor of the opinion score. For example, the word "提高" (raise, positive) is composed of "提" (carry or lift, non-opinion) and "高" (high,

positive). The complement morpheme "高" (high) describes the resulting state of the verb morpheme "提" (raise), so both strength and polarity depend on the morpheme "高" (high).

(6) Negation Type: A negative character specified in a predefined set *NC* has a negation effect on the opinion score of the other character. The strength depends on the modified morpheme while the polarity of the word is the negation of the polarity of the modified morpheme.

$$\begin{aligned}&\text{if } (C_1 \in NC) \text{ then } S(C_1C_2) = (-1)\times S(C_2)\\&\text{else } S(C_1C_2) = (-1)\times S(C_1)\end{aligned} \quad (9)$$

(7) Confirmation Type: A positive character specified in a predefined set *PC* ensures that the opinion score of a word only comes from the other character. Therefore, the opinion score of this word is determined by the modified morpheme.

$$\text{if } (C_1 \in PC) \text{ then} S(C_1C_2) = S(C_2) \text{ else} S(C_1C_2) = S(C_1) \quad (10)$$

(8) Others: Since words of this type contain no clear cues for their morphological structures, we postulate that both characters have the same contribution, and adopt Formula (5).

4 Identification of Morphological Types

To compute the opinion score of a word according to formulae in Section 3, we must know its morphological type from the morphological structure, i.e., the parts of speech of the composite morphemes. Currently, part of speech tagging is performed at the word level rather than the morpheme level, and morpheme-tagging corpus is not available. We consider an on-line Chinese dictionary, Dictionary of Chinese Words by Ministry of Education, Taiwan (*MOEDCW*), as a corpus, and compute the statistics of each morpheme in it.

Two classifiers, *CRF* classifier and *SVM* classifier are proposed to recognize morphological types (1)-(5). Morphological types (6) to (8) are determined by rules such as whether two composite characters are morphemes; whether there are confirmation/negation morphemes; and so on.

4.1 MOEDCW Corpus

MOEDCW corpus provides possible parts of speech for each morpheme by treating it as a unigram word, and possible senses under each part of speech. In each entry, there are a sense definition and some example words. Figures 1 and 2 show the specifications of two morphemes "冒" and "汗". The morpheme "冒" has three parts of speech (verb, adverb and noun) and includes 3, 1, and 1 senses. There are 3, 3, and 2 example words listed under the three verb senses.

We can find the correct parts of speech of the composite characters of a word when it is an example word in the dictionary. However, not all words are listed in the corpus. Consider the word "冒汗" (sweat, verb). Figure 1 shows that "冒汗" (sweat) is an example word listed under the verb sense of the character "冒" (perspire), thus the character "冒" (perspire) in the word "冒汗" (sweat) functions as a verb. However, "冒汗" (sweat) is not an example for the character "汗" (sweat). Figure 2 show that there are two possible parts of speech, noun and verb, for the character "汗" (sweat). We then show how to identify its function in the word "冒汗".

verb	**1**	Goes out from the button to the top or from inside to outside. For example, **fume**, **smoking**, and **sweat**. 由下往上或往外透出、發散。如：「**冒**煙」、「**冒**氣」、「**冒汗**」。
	2	Burst into or regardless of. For example, **take risk**, **to offend**, and **offense**. 衝犯、不顧。如：「**冒**險」、「**冒**犯」、「衝**冒**」。…
	3	Fake or on the pretext of. For example, **personate** and **to pretend to be**. 假稱、假託。如：「**冒**名」、「假**冒**」。
adverb	**1**	Crude or rash. For example, **offensively** and **advance rashly**. 鹵莽、莽撞。如：「**冒**犯」、「**冒**進」。
noun	**1**	Family name. 姓。

Figure 1: Specification of "冒" in MOEDCW

noun	**1**	Sweat. For example, **cold sweat**, **night sweat**, **sweatiness**, and **to drip with sweat**. 由動物皮膚的毛細孔所排泄出的液體。如：「冷**汗**」、「盜**汗**」、「**汗**流浹背」、「揮**汗**如雨」。…
	2	Family name 姓。
verb	**1**	To sweat 流**汗**、使出**汗**。

Figure 2: Specification of "汗" in MOEDCW

$$T(C, POS) = NumberOfSenses(C, POS) \quad (11)$$

The number of possible meanings one character can bear when it functions as a certain part of

speech is employed to estimate how often this part of speech is used. The function $T(C, POS)$ shown in Formula (11) defines the score of a character C functioning as a particular part of speech POS. Here, POS may be noun (N), adjective (ADJ), verb (V), adverb (ADV), auxiliary (AUX), conjunction (CONJ), pronoun (PRON), preposition (PREP), and interjection (INT). In Figure 2, T(汗<sweat>, N) = 2 and T(汗<sweat>, V) = 1.

4.2 Features for Classifiers

Features for training *SVM* and *CRF* classifiers include the pronunciation and the tone of the word, parts of speech of the first and the second characters of training words, and the position information of the composite characters. The tone of the word is acquired from MOEDCW. The parts of speech are estimated by Formula (11). $f(C, POS, k,$ start/end) counts the number of k-grams (k=2, 3, 4). In Figures 1 and 2, f(冒, V, 2, start)=6, f(冒, V, 2, end)= 2, f(冒, ADV, 2, start) = 2, and f(冒, ADV, 2, end)=0. This example shows that when the character "冒" functions as a verb or an adverb, it serves as the starting character more often than the ending character.

4.3 CRF and SVM Classifier

CRF and *SVM* are both common used algorithms for building classifiers (Lafferty *et al.*, 2001). We adopted *CRF++*[1] and *libSVM* (Chang and Lin, 2001) to develop our classifiers. The features for training our *CRF* and *SVM* classifiers include the input word W, the tone of W, the first and the second characters C_1 and C_2, $T(C_1, POS)$, $T(C_2, POS)$, $f(C_1, POS, k,$ start), $f(C_1, POS, k,$ end), $f(C_2, POS, k,$ start), and $f(C_2, POS, k,$ end). POS denotes one of nine parts of speech in *MOEDCW*, and k equals to 2, 3 or 4.

Using *SVM* is straightforward. To classify a word into one of the morphological structure types, we construct the word's feature vector and input the vector into SVM. When using *CRF*, a different approach is taken. When predicting the classes of two successive instances, *CRF* takes the predicted class of the first instance into account when predicting the second instance's class. Here is how we exploit this capability. In a nutshell, we perform classification at the character level instead of the word level. Let W be a word composed of the two characters C_1 and C_2. Let $\boldsymbol{v}$ be the feature vector of W. Let t be the morphological structure type of W. We define C_1's feature vector to be composed of the features in $\boldsymbol{v}$ which are related to C_1, e.g., $T(C_1,$ verb). Similarly, C_2's feature vector is composed of the features in $\boldsymbol{v}$ which are related to C_2. C_1's class and C_2's class are defined as t_1 and t_2, respectively. Since t has five possible values, there are 10 character classes.

To determine a word W's morphological structure type, we first apply *CRF* on W's constituent characters C_1 and C_2's feature vectors. For C_1, *CRF* will return a set of probabilities $P(C_1,t\_q)$, where $q \in \{1, 2\}$, indicating the likelihood of C_1 being an instance of class t_q. Similarly, a set of probabilities $P(C_2,t\_q)$ is returned for C_2. W's morphological structure type is defined as the value of t which maximizes the product of $P(C_1,t\_1)$ and $P(C_2,t\_2)$.

Though *CRF* is mostly used for sequential labeling, the idea of using CRF is to tail this classification questions into a labeling question in order to utilizing the position information of characters. As mentioned, if a word W of two characters *C1C2* is of type 1, *CRF* will label *C1* 1_1 (type1_1st char) and *C2* 1_2 (type1_2nd char). The labeling of each character considers both the previous character's features and the next character's features. That is, if the current character is the first character, its previous character is an empty character (which is used for segmenting sequences in *CRF*); if the current character is the second character, its next character is an empty character. Hence the position information will be considered by *CRF*.

5 Experiments and Discussion

Experiments verify whether the morphological types benefit opinion polarity detection on words. The relation between the performance of morphological classifiers and opinion polarity detection is discussed.

5.1 Experimental Setup

To compare the bag-of-characters approach (Ku *et al.*, 2007) with our morphological structure approach, we adopt the same evaluation data set containing 836 words. To evaluate the performance of our two morphological classifiers, we prepare two sets of words, including the testing set of 836 words for word-level opinion prediction (abbreviated as *OP*), and a set of 8,186 words selected from words in *MOEDCW* corpus and news documents except those can be classi-

[1] http://crfpp.sourceforge.net/

fied by patterns (abbreviated as *TRAIN set*), all with their morphological types annotated. Table 1 lists the distributions of morphological types in *OP* and *TRAIN* sets.

The polarity of words is predicted by their opinion scores ranging between -1 to 1. We set a positive threshold. Those words with scores above it are considered as positive while those below this threshold multiplied by (-1) are regarded as negative. The words with non-zero scores falling between the positive and negative thresholds are neutral. Fifty grids from 0 to 0.5 are searched for the best threshold. Since the opinion extraction at word level concerns only word structure, no retraining for the best threshold is need when domain shifts, which is a superiority of our method.

5.2 Morphological Type Classification and Polarity Detection

The performances of *CRF* and *SVM* classifiers on each morphological type are listed in Table 2. We perform four-fold cross validation on the *TRAIN* set. Results show that *CRF* classifier achieves better performance than *SVM* classifier in this task. The accuracy of *CRF* classifier (0.70) is 8% higher than that of *SVM* classifier (0.62). Note those type 8 words which could be extracted by rules are excluded from classification experiment. The remaining type 8 words are usually proper names. It is difficult for both classifiers to identify such words.

Table 3 further shows the performance of polarity prediction using morphological types determined by *CRF* classifier and *SVM* classifier. The performance of polarity detection is evaluated by the f-score defined in Formula (12).

The f-scores of polarity detection using *CRF* classified types and *SVM* classified types are 0.5806 and 0.5938, respectively. Both of them outperform baseline's f-score 0.5455, i.e., the bag-of-characters approach (Ku *et al.*, 2007). Experiments show that adopting morphological types annotated by two classifiers for polarity prediction has little difference. In other words, *CRF* and *SVM classifiers* have an 8% f-score difference in their best performance of classification, while the performance gap in word polarity prediction using morphological types provided by these two classifiers is around 1.3% only (0.5806 vs. 0.5938). The reason may be that we define scoring functions of each morphological type in a straightforward way. If they are not the best scoring functions, the benefit of considering the morphological type information could be restricted. Nevertheless, experimental results show that morphological type information is useful for word polarity detection (with p-value less than 0.05).

$$P = \frac{correct(opinion) \cap correct(polarity)}{proposed(opinion)},$$
$$R = \frac{correct(opinion) \cap correct(polarity)}{gold(opinion)}, \quad (12)$$
$$f-score = \frac{2 \cdot P \cdot R}{P + R}.$$

set/type	1	2	3	4	5	6	7	8
TRAIN	26.15	44.97	1.64	15.14	9.22	0	0	2.88
OP	45.8	24.4	1.3	7.9	8.0	2.3	0.5	9.8

Table 1: The Percentage of distribution for morphological types in *TRAIN* and *OP* sets

MorphoType	1	2	3	4	5	8	Accuracy
CRF	0.63	0.78	0.41	0.66	0.78	0.17	0.70
SVM	0.49	0.73	0.22	0.52	0.55	0	0.62

Table 2: The f-score of *CRF* and *SVM* classifiers

We further examine how well our polarity detection method works in combination with a word sentiment dictionary. We use the NTUSD[2] word sentiment dictionary. If a word appears in NTUSD, then the word's polarity is the one specified in NTUSD. If a word does not appear in NTUSD, then the word's polarity is determined using our morphological type method.

After introducing a sentiment dictionary NTUSD[3], *CRF* and *SVM* classifiers both achieve the f-score 0.77 for opinion word extraction, and achieve f-scores 0.61 and 0.62 for polarity detection, respectively. Note that if only NTUSD is used to extract opinion words by string matching, the f-score is only 0.44.

[2] http://nlg18.csie.ntu.edu.tw:8080/opinion/

[3] http://nlg18.csie.ntu.edu.tw:8080/opinion/

Polarity f-score	Without NTUSD	With NTUSD
Ku	0.5455	0.5789
CRF type	0.5806	0.6100
SVM type	0.5938	0.6246

Table 3: Prediction with Morphological Types

We further analyze the improvement of polarity prediction for each morphological type. We find that the f-scores of polarity prediction of all morphological types are improved in different degrees, and among them the performance of type 2 words are improved the most. We have shown that our method can assign an opinion score to an arbitrary word without any word thesauri by considering its morphological information. Moreover, since the Substantive-Modifier (type 2) is the most common way to form a new word in the Chinese language (Cheng and Tian, 1992), the result presents the strength of our method in solving the coverage problem.

6 Syntactic Structure for Chinese Opinion Analysis

As mentioned, the relations introduced in Section 2 exist not only within words, but also between sentence segments. Relations between sentence segments are represented by structural trios hereafter and will be introduced in next section. We have already shown that morphological types are useful when extracting opinion words and would like to further testify whether structural trios also benefit the opinion analysis on sentences. We annotate these relations manually, propose a method to identify these relations, and compare results of experimental settings using structural trios with those not using structural trios.

6.1 Structural Trio

Each node in a parsing tree dominates a word string in a sentence. Linguistics have shown that there are also five relations between sentence segments: Parallel, Substantive-Modifier, Subjective-Predicate, Verb-Object, and Verb-Complement, same as morphological types (1) to (5). Because parsing trees have hierarchical structures, we define a structural trio to represent a relation between two nodes as follows:

(1) A structure trio contains two children nodes which bear a relation.

(2) A structure trio contains one head node which is the nearest common parent of two children nodes in (1).

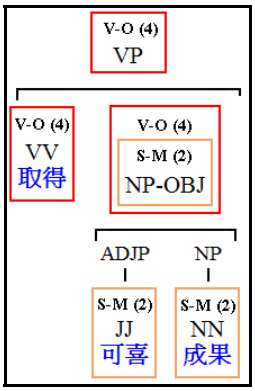


Figure 3: Example of structural trios

Figure 3 shows an example of a structure trio. It is a part of a parsing tree containing words "取得" (obtain), "可喜" (happy), "成果" (results). Two structural trios are shown in this example. The lower one contains two children nodes "可喜" (happy) and "成果" (results), and is labeled as Substantive-Modifier (S-M (2)) in their nearest common parent node, while the upper one contains two children nodes "取得" (obtain) and "可喜成果" (happy results), and is labeled as Verb-Object (V-O (4)).

6.2 Experimental Corpus

To experiment with structural trios, we need the parsing trees of all experimental sentences. For this purpose, we adopted Chinese Treebank 5.1[4] as the experimental materials. Chinese Treebank contains raw Chinese news documents together with their segmented, part of speech tagged, and parsed versions. The parsed documents are adopted in experiments utilizing structural trios, and the part of speech tagged documents are used in experiments not utilizing structural trios.

In Chinese Treebank, a unique ID is labeled on each sentence. For each sentence, we had three annotators label their opinions and then we generate the gold standard following NTCIR[5] MOAT protocol (Seki *et al.*, 2008). We also annotated structure trios in Chinese Treebank. A total of 17,159 sentences are obtained after dropping some faulty sentences such as empty sentences and sentences composed of more than one parsing tree. The statistics of opinion sentences and structural trios in the constructed experimental materials are shown in Table 4 and Table 5.

[4] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T01

[5] http://research.nii.ac.jp/ntcir/index-en.html

	Opinion			Non-Opinion
	Positive	Neutral	Negative	
#	6,380	1,537	1,714	7,528
	9,631			
%	66.24	15.96	17.80	43.87
	56.13			

Table 4: Statistics of opinion sentences

Trio Type	Number	Percentage %
2	18,483	36.85
3	13,687	27.29
4	15,970	31.84
5	965	1.92
Others	1,054	2.10
Total	50,159	100.00

Table 5: Statistics of structural trios

6.3 Experiment Setup

The aim of our experiments is to know how opinion analysis approach performs when morphological and syntactic structures are incorporated. They are compared with the bag-of-character and bag-of-word approaches. We implemented the bag-of-word approach proposed by Ku *et al.* (2007) to show its performance on Chinese Treebank. In their approach, the opinion scores of words are summed to generate the opinion scores of sentences, and the negation words will negate the closest opinion words. Based on this approach, we further consider structural trios to experiment whether syntactic structures of sentences are beneficial for opinion analysis. Because the scoring functions may not be straight forward as those we have adopted for opinion word extraction, we did not design scoring functions for utilizing all types of structural trios. Instead, we emphasize their original opinion scores by multiplying a variable alpha to see whether these structures are important. In this paper, alpha equals five.

We have shown that word morphological structures benefit the word opinion extraction. When we experiment on sentences, we also incorporate the word morphological structures to see whether they are also useful for opinion analysis on sentences. Five experimental settings are listed as below:

(1) bag[w]-bag[s]: structural information is not considered for both words and sentences. The bag-of-character approach is used to calculate the opinion scores of words, and the bag-of-word approach sentences.

(2) struc[w]-bag[s]: morphological structures are utilized to calculate word opinion scores, but structural trios are not considered. The bag-of-word approach is used to calculate the opinion scores of sentences.

(3) bag[w]-struc[s]: structural trios are considered for calculating sentence opinion scores, while the bag-of-character approach is used to calculate the opinion scores of words.

(4) struc[w]-(m)struc[s]: both word morphological structures and manually labeled structural trios are adopted.

(5) struc[w]-struc[s]: both morphological structure of words and system labeled structural trios are adopted.

As we have shown that NTUSD is beneficial to the opinion analysis at word level, it is used as described in section 5.2 by default.

Our system adopted *CRF* algorithm to label structural trios for setting (5). The content string and the part of speech of the current node, its parent node, its offspring nodes in the next three generations, together with the depth of the current node in the Chinese Treebank, are used as the features for each node in *CRF*. The co-occurrence of the current node and all its siblings are defined in *CRF*'s template file. *CRF* will label whether the current node is the first child or the second child of a certain relation in a structural trio, or it is not part of any structural trios. A four-fold experiment is performed for the learning and testing of this labeling process by *CRF*.

6.4 Results and Discussion

Table 6 shows the statistics of manually labeled structural trios in Chinese Treebank and identification performance of *CRF*. Table 7 shows the performance of five experiment settings described in Section 6.3. The experiment results show that the morphological structures of words do not have a large contribution for opinion sentence analysis (setting 1 vs. setting 2; setting 3 vs. setting 4). However, considering the structural trios improve the performance.

Trio Type	Number	Percentage	f-Score
2	18,483	36.85%	0.4883
3	13,687	27.29%	0.4944
4	15,970	31.84%	0.6360
5	965	1.92%	0.2034
Others	1,054	2.10%	
Total	50159	100%	

Table 6: Statistics and Results of Identifying Structural Trios

Setting	Word [w]	Sentence [s]	f-Score (opinion)	f-Score (polarity)
1	bag	bag	0.7073	0.4988
2	struc	bag	0.7162	0.5117
3	bag	struc	0.8000	0.5361
4	struc	(m)struc	0.7922	0.5297
5	struc	struc	0.7993	0.5187

Table 7: Results of Opinion Extraction on Chinese Treebank

By summarizing the experimental results in Section 5 and this section, we can conclude that considering the word morphological structures benefits the opinion polarity detection, but in the current approach its assistance to words does not propagate to sentences. Considering the syntactic structures, however, do help in opinion analysis both for the opinion sentence extraction and the polarity detection. The performance of opinion extraction boosts to an f-score 0.80 and the performance of polarity detection an f-score 0.54.

However, the utilization of structure trios needs the parsing tree of sentences as the prior knowledge. Hence these two kinds of structural information may be suitable for different applications: structural trios for well written sentences such as those in the news articles, while the morphological structures for casually written sentences such as those appear in SMS messages or articles with limit length on the Web.

Because there are no opinion experiments performed on Chinese Treebank, we mention the performance of Ku's approach (setting (1)) for opinion sentence extraction, f-score 0.6846, in NTCIR-7 MOAT task, on news articles, as a result for comparison. Their approach was ranked the second in this task, and the best team achieved an f-score 0.7453.

7 Conclusion and Future Work

This paper considers morphological and syntactic structures in analyzing Chinese opinion words and sentences. For morphological structures, eight Chinese morphological types are defined. *CRF* classifier and *SVM* classifier for morphological type classification are proposed. Experiments show that *CRF* classifier achieves the best accuracy 0.70 in type classification, which is 8% better than *SVM* classifier. We further show that word morphological structures benefit the opinion word extraction significantly. With the help of the sentiment dictionary NTUSD, the f-score of opinion word extraction achieves 0.77 and the f-score of the word polarity detection achieves 0.62 when the word morphological types are provided by the *SVM* classifier. They are comparably better than bag-of-character approach and the dictionary based approach.

We defined structural trios to represent the relations between sentence segments and also extract these relations using *CRF* algorithm. Results show that considering structural trios benefits the opinion analysis on sentences. An f-score 0.80 for opinion extraction and an f-score 0.54 for polarity detection are achieved, which is a great improvement.

The opinion scoring functions for morphological types and structural trios are critical for polarity detection, and scoring functions for words determine the scoring functions for sentences. Now we define these functions intuitively based on linguistic rules, but learning methods like regression will be investigated in the future. Examining the interaction of cues from word and sentence levels on the opinion sentence extraction and the opinion polarity detection is our next goal.

Acknowledgement

Research of this paper was partially supported by National Science Council, Taiwan, under the contract NSC95-2221-E-002-265-MY3.

References

Banea, C., Mihalcea, R., Wiebe, J. and Hassan, S. 2008. Multilingual Subjectivity Analysis Using Machine Translation. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP 2008)*.

Bautin, M., Vijayarenu, L. and Skiena, S. 2008. International sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*.

Carenini, G., Ng, R. T. and Pauls, A. 2006. Interactive Multimedia Summaries of Evaluative Text. In *Proceedings of the 11th International Conference on Intelligent User Interfaces* (pp. 124-131), Sydney, Australia.

Cesarano, C., Picariello, A., Reforgiato, D. and Subrahmanian, V.S. 2007. The OASYS 2.0 Opinion Analysis System. Demo in *Proceedings of International Conference on Weblogs and Social Media* (pp. 313-314), Boulder, CO USA.

Chang, Chih-Chung and Lin, Chih-Jen. 2001. LIBSVM: a library for support vector machines, http://www.csie.ntu.edu.tw/~cjlin/libsvm

Chen, A., Xu, L., Gey, F.C. and Meggs, J. 1997. Chinese Text Retrieval without Using a Dictionary. *ACM SIGIR Forum*, Volume 31, Issue SI (pp. 42-49).

Cheng, X.-H. and Tian, X.-L. 1992. *Modern Chinese*. Bookman Books Ltd.

Dave, K., Lawrence, S., and Pennock, D.M. 2003. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. In *Proc. of the 12th International WWW Conference* (pp. 519-528).

Kawai, Y., Kumamoto, T. and Tanaka, K. 2007. Fair News Reader: Recommending news articles with different sentiments based on user preference. In *Proceedings of Knowledge-Based Intelligent Information and Engineering Systems (KES), No. 4692 in Lecture Notes in Computer Science* (pp. 612–622).

Kim, S.-M. and Hovy, E. 2004. Determining the Sentiment of Opinions. In *Proc. of the 20th ICCL* (pp. 1367-1373).

Ku, L.-W. and Chen, H.-H. 2007. Mining Opinions from the Web: Beyond Relevance Retrieval. *Journal of American Society for Information Science and Technology*, Special Issue on Mining Web Resources for Enhancing Information Retrieval, 58(12), 1838-1850.

Lafferty, J., McCallum, A. and Pereira, F. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, In *Proc. of ICML* (pp.282-289).

Pang, B., Lee, L. and Vaithyanathan, S. 2002. Thumbs up? Sentiment Classification Using Machine Learning Techniques. In *Proc. of the 2002 Conference on EMNLP* (pp. 79-86).

Riloff, E. and Wiebe, J. 2003. Learning Extraction Patterns for Subjective Expressions. In *Proc. of the 2003 Conference on EMNLP* (pp. 105-112).

Seki, Y., Evans, D. K., Ku, L.-W., Sun, L., Chen, H.-H. and Kando, N. 2008. Overview of Multilingual Opinion Analysis Task at NTCIR-7. In *Proceedings of the 7th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering, and Cross-Lingual Information Access.*

Somasundaran, S., Ruppenhofer, J. and Wiebe, J. 2007. Detecting arguing and sentiment in meetings. *Proceedings of the SIGdial Workshop on Discourse and Dialogue*, 2007.8.6

Takamura, H., Inui, T. and Okumura, M. 2005. Extracting Semantic Orientations of Words Using Spin Model. In *Proc. of the 43rd Annual Meeting of the ACL* (pp. 133-140).

Tzeng, H. and Chen, K.-J. 2002. Design of Chinese Morphological Analyzer. In *Proc. of the 1st SIGHAN Workshop on Chinese Language Processing*, vol.18, 1-7.

Wiebe, J. 2000. Learning Subjective Adjectives from Corpora. In *Proc. of the 17th National Conference on AAAI and Twelfth Conference on IAAI* (pp. 735-740).

Finding Short Definitions of Terms on Web Pages

Gerasimos Lampouras[*] and **Ion Androutsopoulos**[*+]
[*]Department of Informatics, Athens University of Economics and Business, Greece
[+]Digital Curation Unit, Research Centre "Athena", Athens, Greece

Abstract

We present a system that finds short definitions of terms on Web pages. It employs a Maximum Entropy classifier, but it is trained on automatically generated examples; hence, it is in effect unsupervised. We use ROUGE-W to generate training examples from encyclopedias and Web snippets, a method that outperforms an alternative centroid-based one. After training, our system can be used to find definitions of terms that are not covered by encyclopedias. The system outperforms a comparable publicly available system, as well as a previously published form of our system.

1 Introduction

Definitions of terms are among the most common types of information users search for on the Web. In the TREC 2001 QA track (Voorhees, 2001), where the distribution of question types reflected that of real user logs, 27% of the questions were requests for definitions (e.g., "What is gasohol?", "Who was Duke Ellington?"). Consequently, some Web search engines provide special facilities (e.g., Google's "define:" query prefix) that seek definitions of user-specified terms in on-line encyclopedias or glossaries; to save space, we call both "encyclopedias". There are, however, often terms that are too recent, too old, or less widely used to be included in encyclopedias. Their definitions may be present on other Web pages (e.g., newspaper articles), but they may be provided indirectly (e.g., "He said that *gasohol, a mixture of gasoline and ethanol*, has been great for his business.") and they may be difficult to locate with generic search engines that may return dozens of pages containing, but not defining the terms.

We present a system to find short definitions of user-specified terms on Web pages. It can be used as an add-on to generic search engines, when no definitions can be found in on-line encyclopedias. The system first invokes a search engine using the (possibly multi-word) term whose definition is sought, the *target term*, as the query. It then scans the top pages returned by the search engine to locate 250-character snippets with the target term at their centers; we call these snippets *windows*. The windows are candidate definitions of the target term, and they are then classified as acceptable (positive class) or unacceptable (negative class) using supervised machine learning. The system reports the windows for which it is most confident that they belong in the positive class. Table 1 shows examples of short definitions found by our system. In our experiments, we allow the system to return up to five windows per target term, and the system's response is counted as correct if any of the returned windows contains an acceptable short definition of the target. This is similar to the treatment of definition questions in TREC 2000 and 2001 (Voorhees, 2000; Voorhees, 2001), but the answer is sought on the Web, not in a given document collection of a particular genre.

More recent TREC QA tracks required definition questions to be answered by lists of complementary text snippets, jointly providing required or optional information nuggets (Voorhees, 2003). In contrast, we focus on locating single snippets that include self-contained short definitions. Despite its simpler nature, we believe the task we address is of practical use: a list of single-snippet definitions from Web pages accompanied by the source URLs is a good starting point for users seeking definitions of terms not covered by encyclopedias. We also note that evaluating multi-snippet definitions can be problematic, because it is often difficult to agree which information nuggets should be treated as required, or even optional (Hildebrandt et al., 2004). In contrast, earlier experimental results we have reported (Androutsopoulos and Galanis, 2005) show strong inter-assessor agreement ($K > 0.8$) for single-snippet definitions (Eugenio and Glass, 2004). The task we address also differs from DUC's query focused summarization (Dang, 2005; Dang, 2006). Our queries are single terms, whereas DUC queries are longer topic

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1270–1279,
Singapore, 6-7 August 2009.

Target term: Babesiosis
(...) Babesiosis is a rare, severe and sometimes fatal tick-borne disease caused by various types of Babesia, a microscopic parasite that infects red blood cells. In New York state, the causative parasite is babesia microti. Who gets Babesiosis? Babesiosis (...)
Target term: anorexia nervosa
(...) anorexia nervosa is an illness that usually occurs in teenage girls, but it can also occur in teenage boys, and adult women and men. People with anorexia are obsessed with being thin. They lose a lot of weight and are terrified of gaining weight. The (...)
Target term: Kinabalu
(...) one hundred and thirty eight kilometers from Kota Kinabalu, the capital of the Malaysian state of Sabah, rises the majestic mount Kinabalu. With its peak at 4,101 meters (and growing), mount Kinabalu is the highest mountain in south-east Asia. This (...)
Target term: Pythagoras
(...) Pythagoras of Samos about 569 BC - about 475 BC click the picture above to see eleven larger pictures Pythagoras was a Greek philosopher who made important developments in mathematics, astronomy, and the theory of music. The theorem now known as (...)
Target term: Sacajawea
(...) Sacajawea was a Shoshone Indian princess. The Shoshone lived from the rocky mountains to the plains. They lived primarily on buffalo meat. The shoshone traveled for many days searching for buffalo. They hunted on horseback using the buffalo for food (...)
Target term: tale of Genji
(...) the tale of Genji This site aims to promote a wider understanding and appreciation of the tale of Genji - the 11th century Japanese classic written by a Heian court lady known as Murasaki Shikibu. It also serves as a kind of travel guide to the world (...)
Target term: Jacques Lacan
(...) who is Jacques Lacan? John Haber in New York city a primer for pre-post-structuralists Jacques Lacan is a Parisian psychoanalyst who has influenced literary criticism and feminism. He began work in the 1950s, in the Freudian society there. It was a (...)

Table 1: Definitions found by our system.

descriptions, often entire paragraphs; furthermore, we do not attempt to compose coherent and cohesive summaries from several snippets.

The system we present is based on our earlier work (Miliaraki and Androutsopoulos, 2004), where an SVM classifier (Cristianini and Shawe-Taylor, 2000) was used to separate acceptable windows from unacceptable ones; the SVM also returned confidence scores, which were used to rank the acceptable windows. On datasets from the TREC 2000 and 2001 QA tracks, our earlier system clearly outperformed the methods of Joho and Sanderson (2000; 2001) and Prager et al. (2001; 2002), as reported in previous work (Miliaraki and Androutsopoulos, 2004). To train the SVM, however, thousands of training windows were required, each tagged as a positive or negative example. Obtaining large numbers of training windows is easy, but manually tagging them is very time-consuming. In the TREC 2000 and 2001 datasets, it was possible to tag the training windows automatically by using training target terms and accompanying regular expression patterns provided by the TREC organizers. The regular expressions covered all the known acceptable definitions of the corresponding terms that can be extracted from the datasets. When the training windows, however, are obtained from the Web, it is impossible to construct manually regular expressions for all the possible phrasings of the acceptable definitions in the training windows.

In subsequent work (Androutsopoulos and Galanis, 2005), we developed ATTW (automatic tagging of training windows), a technique that produces arbitrarily large collections of training windows from the Web with practically no manual effort, in effect making our overall system unsupervised. ATTW uses training terms for which several encyclopedia definitions are available, and compares each Web training window (each window extracted from the pages the search engine returned for a training term) to the corresponding encyclopedia definitions. Web training windows that are very similar (or dissimilar) to the corresponding encyclopedia definitions are tagged as positive (or negative) examples; if the similarity is neither too high nor too low, the window is not included in the classifier's training data. Previously reported experiments (Androutsopoulos and Galanis, 2005) showed that ATTW leads to significantly better results, compared to training the classifier on all the available TREC windows, for which regular expressions are available, and then using it to classify Web windows.

Note that in ATTW the encyclopedia definitions are used only during training. Once the classifier has been trained, it can be used to discover definitions on arbitrary Web pages. In fact, during testing we discard windows originating from on-line encyclopedias, simulating the case where we seek definitions of terms not covered by encyclopedias; we also ignore windows from on-line encyclopedias during training. Also, note that the classifier is trained on Web windows, not directly on encyclopedia definitions, which allows it to avoid relying excessively on phrasings that are common in encyclopedia definitions, but uncommon in more indirect definitions of arbitrary Web pages. Fur-

thermore, training the classifier directly on encyclopedia definitions would not provide negative examples.

In our previous work with ATTW (Androutsopoulos and Galanis, 2005) we used a measure constructed by ourselves to assess the similarity between Web windows and encyclopedia definitions. Here, we use the more established ROUGE-W measure (Lin, 2004) instead. ROUGE-W and other versions of ROUGE have been used in summarization to measure how close a machine-authored summary is to multiple human summaries of the same input. We use ROUGE-W in a similar setting, to measure how close a training window is to multiple encyclopedia definitions of the same term. A further difference from our previous work is that we also use ROUGE-W when computing the features of the windows to be classified. Previously, the SVM relied, among others, on Boolean features indicating if the target term was preceded or followed in the window to be classified by a particular phrase indicating a definition (e.g., "target, a kind of", "such as target"). The indicative phrases are selected automatically during training, but now the corresponding features are not Boolean; their values are the ROUGE-W similarity scores between an indicative phrase and the context of the target term in the window. This allows the system to soft-match the phrases to the windows (e.g., encountering "target, another kind of", instead of "target, a kind of").[1]

In our new system we also use a Maximum Entropy (MAXENT) classifier (Ratnaparkhi, 1997) instead of an SVM, because much faster implementations of the former are available.[2] We present experimental results showing that our new system significantly outperforms our previously published one. The use of the MAXENT classifier by itself improved slightly our results, but the improvements come mostly from using ROUGE-W.

Apart from presenting an improved version of our system, the main contribution of this paper is a detailed experimental comparison of our new system against Cui et al.'s (2004; 2005; 2006; 2007). The latter is particularly interesting, because it is well published, it includes both an alternative, centroid-based technique to automatically tag training examples and a soft-matching classifier, and it is publicly available.[3] We show that ATTW outperforms Cui et al.'s centroid-based technique, and that our overall system is also clearly better than Cui et al.'s in the task we address.

Section 2 discusses ATTW with ROUGE-W, Cui et al.'s centroid-based method to tag training examples, and experiments showing that ATTW is better. Section 3 describes our new overall system, the system of Cui et al., and the baselines. Section 4 reports experimental results showing that our system is better than Cui et al.'s, and better than our previously published system. Section 5 discusses related work; and section 6 concludes.

2 Tagging training windows

During both training and testing, for each target term we keep the r most highly ranked Web pages the search engine returns. We then extract the first f windows of the target term from each page, since early occurrences of the target terms on pages are more likely to be definitions. We, thus, obtain $r \cdot f$ windows per term.[4] When testing, we return the k windows of the target term that the classifier is most certain they belong in the positive class. In our experiments, $r = 10$, $f = 5$, $k = 5$. During training, we train the classifier on the $q \cdot r \cdot f$ windows we obtain for q training target terms; in our experiments, q ranged from 50 to 1500. Training requires tagging first the training windows as positive or negative, possibly discarding windows that cannot be tagged automatically.

2.1 ATTW with ROUGE-W similarity

To tag a training window w of a training term t with ATTW and ROUGE-W, we obtain a set C_t of definitions of t from encyclopedias.[5] Stop-words, punctuation, and non-alphanumeric characters are removed from C_t and w, and a stemmer is applied; the testing windows undergo the same preprocessing.[6] For each definition $d \in C_t$, we find the longest common word subsequence of w and d. If w is the word sequence $\langle A, B, F, C, D, E \rangle$

[1] We also experimented with other similarity measures (e.g., edit distance) and ROUGE variants, but we obtained the best results with ROUGE-W.

[2] We use Stanford's classifier; see http://nlp.stanford.edu/.

[3] See http://www.cuihang.com/software.html. The software and a demo of our system, and the datasets we used are also freely available; see http://nlp.cs.aueb.gr/.

[4] We used Altavista in our experiments. We remove HTML tags and retain only the plain text of the pages.

[5] The training terms were randomly selected from the index of http://www.encyclopedia.com/. We used Google's "define:" to obtain definitions from other encyclopedias.

[6] We use the 100 most frequent words of the BNC corpus (http://www.natcorp.ox.ac.uk/) as the stop-list, and Porter's stemmer (http://tartarus.org/∼martin/PorterStemmer/).

and $d = \langle A, B, E, C, G, D \rangle$, the longest common subsequence is $\langle A, B, C, D \rangle$. The longest common subsequence is divided into consecutive matches, producing in our example $\langle A, B|C|D \rangle$. We then compute the following score (weighted longest common subsequence), where m is the number of consecutive matches, k_i is the length of the i-th consecutive match, and f is a weighting function. We use $f(k) = k^a$, where $a > 1$ is a parameter we tune experimentally.

$$WLCS(w, d) = \sum_{i=0}^{m} f(k_i)$$

We then compute the following quantities, where $|\cdot|$ is word length, and f^{-1} is the inverse of f.

$$P(w, d) = f^{-1}(\frac{WLCS(w,d)}{f(|w|)})$$
$$R(w, d) = f^{-1}(\frac{WLCS(w,d)}{f(|d|)})$$
$$F(w, d) = \frac{(1+\beta^2) \cdot R(w,d) \cdot P(w,d)}{R(w,d) + \beta^2 \cdot P(w,d)}$$

In effect, $P(w, d)$ examines how close the longest common substring is to w and $R(w, d)$ how close it is to d. Following Lin (2004), we use $\beta = 8$, assigning greater importance to $R(w, d)$. If $R(w, d)$ is high, the longest common substring is very similar to d; then w (which also includes the longest common substring) intuitively contains almost all the information of d, i.e., all the information of a known acceptable definition (high recall). If $P(w, d)$ is high, the longest common substring is very similar to w; then d (which also includes the longest common substring) contains almost all the information of w, i.e., w does not contain any (redundant) information not included in a known acceptable definition, something we care less for.

The ROUGE-W similarity $sim(w, C_t)$ between w and C_t is the maximum $F(w, d)$, for all $d \in C_t$. Training windows with $sim(w, C_t) > T_+$ are tagged as positive; if $sim(w, C_t) < T_-$, they are tagged as negative; and if $T_- \leq sim(w, C_t) \leq T_+$, they are discarded. We tune the thresholds T_+ and T_- experimentally, as discussed below.

2.2 The centroid-based tagging approach

This method is used in the system of Cui et al. (2004; 2005; 2006; 2007). For each training target term, we construct a "centroid" pseudo-text containing the words that co-occur most frequently with the target term. We then compute the similarity between each training window and the centroid of its target term. If it exceeds a threshold, the window is tagged as positive; Cui et al. produce only positive examples.

The centroid of a training target term t is constructed as follows. For each word u in t's training windows, we compute the centrality score defined below, where SF_t is the number of t's training windows, SF_u is the number of u's windows that can be extracted from the retained Web pages the search engine returned for t, $SF_{t \cap u}$ is the number of windows on the same pages that contain both t and u, and $idf(u)$ is the inverse document frequency of w.[7] Centrality scores are pointwise mutual information with an extra $idf(u)$ factor.

$$centrality(u) = -log(\frac{SF_{t \cap u}}{SF_t + SF_u}) \cdot idf(u)$$

The words u whose centrality scores exceed the mean by at least a standard deviation are added to the centroid of t. Before computing the centrality scores, stop-words, punctuation, and non-alphanumeric characters are removed, and a stemmer is applied, as in ATTW. The similarities between training windows and centroids are then computed using cosine similarity, after turning the centroids and windows into binary vectors that show which words they contain.

2.3 Comparing the tagging approaches

To evaluate the two methods that tag training windows, we selected randomly $q = 200$ target terms, different from those used for training and testing. We collected the $q \cdot r \cdot f = 200 \cdot 10 \cdot 5$ windows from the corresponding Web pages, we selected randomly 400 from the collected 10,000 windows, and tagged them manually as positive or negative.

Figure 1 plots the positive precision of the two methods against their positive recall, and figure 2 shows negative precision against negative recall. For different values of T_+, we obtain a different point in figure 1; similarly for T_- and figure 2. Positive precision is $TP/(TP + FP)$, positive recall is $TP/(TP + FN)$, and likewise for negative precision and recall; TP (true positives) are the positive training windows the method has correctly tagged as positive, FP are the negative windows the method has tagged as positives etc.

For very high (strict) T_+ values, the methods tag very few (or none) training windows as positive; hence, both TP and $TP + FP$ approach (or become) zero; we take positive precision to be zero in that case. Positive recall also approaches (or becomes) zero, which is why both positive recall and

[7] We obtained $idf(u)$ from BNC. Cui et al. use sentences instead of windows, reducing the risk of truncating definitions. We used windows in all systems, to compare fairly.

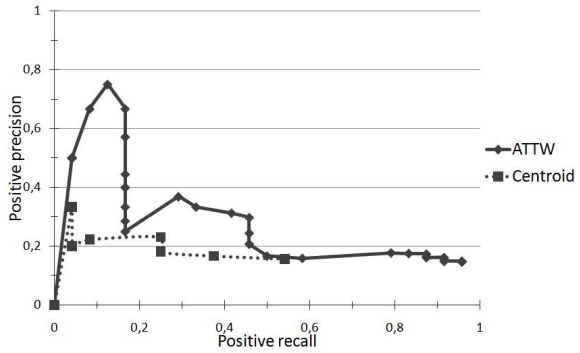


Figure 1: Results of generating positive examples.

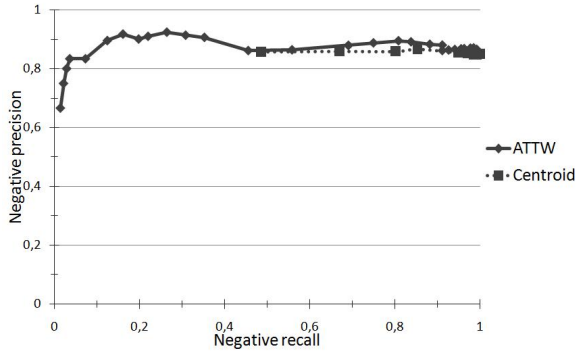


Figure 2: Results of generating negative examples.

precision reach zero in the left of figure 1. Similar comments apply to figure 2, though both methods always tagged correctly at least a few training windows as negative, for the T_- values we tried; hence, negative precision was never zero.

Positive precision shows how certain we can be that training windows tagged as positive are indeed positive; whereas positive recall is the percentage of true positive examples that we manage to tag as such. Figure 1 shows that when using ATTW, we need to settle for a low positive recall, i.e., miss out many positive examples, in order to obtain a reasonably high precision. It also shows that the centroid method is clearly worse when tagging positive examples; its positive precision is almost always less than 0.3. Figure 2 shows that both methods achieve high negative precision and recall; they manage to assign trustworthy negative labels without missing many negative examples. However, ATTW is significantly better when tagging positive examples, as shown in figure 1; hence, it is better than the centroid method.[8]

When using ATTW in practice, we need to select T_+ and T_-. We assign more importance to selecting a T_+ (a point of ATTW's curve in figure 1) that yields high positive precision; the choice of T_- (point in figure 2) is less important, because ATTW's negative precision is always reasonably high. Based on figure 1, we set T_+ to 0.58, which corresponds to positive precision 0.66 and positive recall 0.16. By tuning the two thresholds we can control the number of positively or negatively tagged examples we produce (and their ratio), and the number of examples we discard. Having set T_+, we set T_- to 0.30, a value that maintains the ratio of truly positive to truly negative windows of the 400 manually tagged windows (0.2 to 1), since this is approximately the ratio the classifier will confront during testing; we also experimented with a 1 to 1 ratio, but the results were worse. This T_- value corresponds negative precision 0.70 and negative recall 0.02. Thus, both positive and negative precision is approximately 0.7, which means that approximately 30% of the tags we assign to the examples are incorrect. Our experiments, however, indicate that the classifier is able to generalize well over this noise.

3 Finding new definitions

We now present our overall system, the system of Cui et al., and the baselines.

3.1 Our system

Given a target term, our system extracts $r \cdot f = 10 \cdot 5$ windows from the pages returned by the search engine, and uses the MAXENT classifier to separate them into acceptable and unacceptable definitions.[9] It then returns the $k = 5$ windows the classifier is most confident they are acceptable. The classifier is trained on windows tagged as positive or negative using ATTW. It views each window as a vector of the following features:[10]

SN: The ordinal number of the window on the page it originates from (e.g., second window of the target term from the beginning of the page). Early mentions of a term are more likely to define it.

RK: The ranking of the Web page the window originates from, as returned by the search engine.

[8] We tried different values of ROUGE-W's a parameter in the interval $(1, 2]$. We use $a = 1.4$, which was the value with the best results on the 400 windows. We did not try $a > 2$, as the results were declining as a approached 2.

[9] We do not discuss MAXENT classifiers, since they are a well documented in the literature.

[10] SN and WC originate from Joho and Sanderson (2000).

WC: We create a simple centroid of the window's target term, much as in section 2.2. The centroid's words are chosen based on their frequency in the $r \cdot f$ windows of the target term; the 20 most frequent words are chosen. WC is the percentage of the 20 words that appear in the vector's window.

Manual patterns: 13 Boolean features, each signaling if the window matches a different manually constructed lexical pattern (e.g., "target, a/an/the", as in "Tony Blair, the British prime minister"). The patterns are those used by Joho and Sanderson (2000), and four more introduced in our previous work (Androutsopoulos and Galanis, 2005) and (Miliaraki and Androutsopoulos, 2004). They are intended to perform well across text genres.

Automatic patterns: m numeric features, each showing the degree to which the window matches a different automatically acquired lexical pattern. The patterns are word n-grams ($n \in \{1, 2, 3\}$) that must occur directly before or after the target term (e.g., "*target* which is"). The patterns are acquired as follows. First, all the n-grams directly before or after any target term in the training windows are collected. The n-grams that have been encountered at least 10 times are candidate patterns. From those, the m patterns with the highest precision scores are retained, where precision is the number of positive training windows the pattern matches over the total number of training windows it matches; we use $m = 300$ in our experiments, based on the results of our previous work. The automatically acquired patterns allow the system to detect definition contexts that are not captured by the manual patterns, including genre-specific contexts. The value of each feature is the ROUGE-W score between a pattern and the left or right context of the target term in the window.

3.2 Cui et al.'s system

Given a target term t, Cui et al. (2004; 2005; 2006; 2007) initially locate sentences containing t in relevant documents. We use the $r \cdot f = 10 \cdot 5$ windows from the pages returned by the search engine, instead of sentences. Cui et al. then construct the centroid of t, and compute the cosine similarity of each one of the $r \cdot f$ windows to the centroid, as in section 2.2. The 10 windows that are closer to the centroid are considered candidate answers. All candidate answers are then processed by a part-of-speech (POS) tagger and a chunker. The words of the centroid are replaced in all the candidate answers by their POS tags; the target term, noun phrases, forms of the verb "to be", and articles are replaced by special tags (e.g., TARGET, NP), while adjectives and adverbs are removed. The candidate answers are then cropped to L tokens to the left and right of the target term, producing two subsequences (left and right) per candidate answer; we set $L = 3$, which is Cui et al.'s default.

Cui et al. experimented with two approaches to rank the candidate answers, called Bigram Model and Profile Hidden Markov Model (PHMM). Both are learning components that produce soft patterns, though PHMM is much more complicated. In their earlier work, Cui et al. (2005) found the Bigram Model to perform better than PHMM; in more recent experiments with more data (Cui, 2006; Cui et al., 2007) they found PHMM to perform better, but the difference was not statistically significant. Given these results and the complexity of PHMM, we experimented only with the Bigram Model.

In the Bigram Model, the left and right subsequences of each candidate answer are considered separately. Below $S_1, \ldots, S_L$ refer to the slots (word positions) of a (left or right) subsequence, and $t_1, \ldots, t_L$ to the particular words in the slots. For each subsequence $\langle S_1 = t_1, \ldots, S_L = t_L \rangle$ of a candidate answer, we first estimate:

$$P(t_i|S_i) = \frac{|S_i(t_i)| + \delta}{\sum_{t'} |S_i(t_i)| + \delta \cdot N}$$

$$P(t_i|t_{i-1}) = \frac{|S_i(t_i) \wedge S_{i-1}(t_{i-1})|}{|S_i(t_i)|}$$

$P(t_i|S_i)$ is the probability that t_i will appear in slot S_i of a left or right subsequence (depending on the subsequence considered) of an acceptable candidate answer. $P(t_i|t_{i-1})$ is the probability that t_i will follow t_{i-1} in a (left or right) subsequence of an acceptable candidate answer. Cui et al. use only positive training examples, generated by the centroid-based approach of section 2.2. $|S_i(t_i)|$ is the number of times t_i appeared in S_i in the (left or right) subsequences of the training examples. t' ranges over all the words that occurred in S_i in the training examples. $|S_i(t_i) \wedge S_{i-1}(t_{i-1})|$ is the number of times t_i and t_{i-1} co-occurred in the corresponding slots in the training examples. N is the number of different words that occurred in the (left or right) training subsequences, and δ is a constant set to 2, as in Cui et al.'s experiments. Following Cui et al., if t_i is a POS or other special tag then the probabilities above are estimated by counting

only the tags of the training examples. Similarly, if t_i is an actual word, only the actual words (not tags) of the training examples are considered.

The probability of each subsequence could then be estimated as:

$$P(t_1, \ldots, t_L) = P(t_1|S_1) \cdot \prod_{i=2}^{L} (\lambda \cdot P(t_i|t_{i-1}) + (1 - \lambda) \cdot P(t_i|S_i))$$

Instead, Cui et al. use the following scoring measure, which also accounts for the fact that some subsequences may have length $l < L$. They tune λ by Expectation Maximization.

$$P_{norm}(t_1, \ldots, t_L) = \frac{1}{l} \cdot [\log P(t_1|S_1) + \sum_{i=2}^{L} \log(\lambda \cdot P(t_i|t_{i-1}) + (1 - \lambda) \cdot P(t_i|S_i))]$$

The overall score of a candidate answer is then:

$$P = (1 - \alpha) \cdot P_{norm}(left) + \alpha \cdot P_{norm}(right)$$

Again, Cui et al. tune a by Expectation Maximization. Instead, we tuned λ and α by a grid search in $[0, 1] \times [0, 1]$, with step 0.1 for both parameters. For the tuning, we trained Cui et al.'s system on 2,000 randomly selected target terms, excluding terms used for other purposes. We used 160 manually tagged windows to evaluate the system's performance with the different values of λ and α; the 160 windows were selected randomly from the 10,000 windows of section 2.3, after excluding the 400 manually tagged windows of that section. The resulting values for λ and α were 0.7 and 0.6, respectively. Apart from the modifications we mentioned, we use Cui et al.'s original implementation.

3.3 Baseline methods

The first baseline selects the first window of each one of the five highest ranked Web pages, as returned by the search engine, and returns the five windows. The second baseline returns five windows chosen randomly from the $r \cdot f = 10 \cdot 5$ available ones. The third baseline (centroid baseline) creates a centroid of the $r \cdot f$ windows, as in section 2.2, and returns the five windows with the highest cosine similarity to the centroid.[11]

[11] We also reimplemented the definitions component of Chu-Carroll et al. (2004; 2005), but its performance was worse than our centroid baseline.

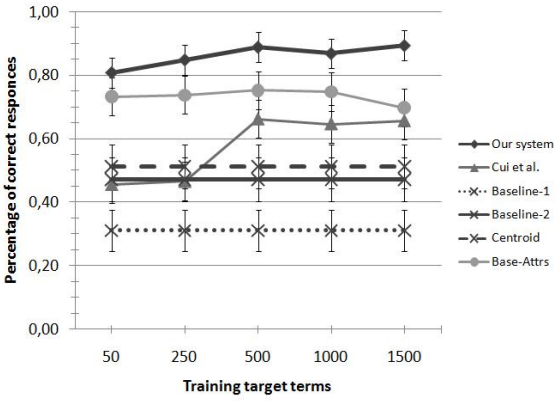


Figure 3: Correct responses, 5 answers/question.

4 Evaluation of systems

We used q training target terms in the experiments of this section, with q ranging from 50 to 1500, and 200 testing terms, with no overlap between training and testing terms, and excluding terms that had been used for other purpose.[12] We had to use testing terms for which encyclopedia definitions were also available, to judge the acceptability of the systems' responses, since many terms are highly technical. We discarded, however, windows extracted from encyclopedia pages when testing, simulating the case where the target terms are not covered by encyclopedias.

As already mentioned, for each target term we extract $r \cdot f = 10 \cdot 5$ windows (or fewer, if fewer are available) from the pages the search engine returns. We then provide these windows to each of the systems, allowing them to return up to $k = 5$ windows, ordered by decreasing confidence. If any of the k windows contains an acceptable short definition of the target term, as judged by a human evaluator, the system's response is counted as correct. We also calculate the Mean Reciprocal Rank (MRR) of each system's responses, as in the TREC QA track: if the first acceptable definition of a response is in the j-th position ($1 \leq j \leq k$), the response's score is $1/j$; MRR is the mean of the responses' scores, i.e., it rewards systems that return acceptable definitions higher in their responses.

Figures 3 and 4 show the results of our experiments as percentage of correct responses and MRR, respectively; the error bars of figure 3 correspond to 95% confidence intervals. Our system clearly outperforms Cui et al.'s, despite the fact that the

[12] The reader is reminded that all terms were selected randomly from the index of an on-line encyclopedia.

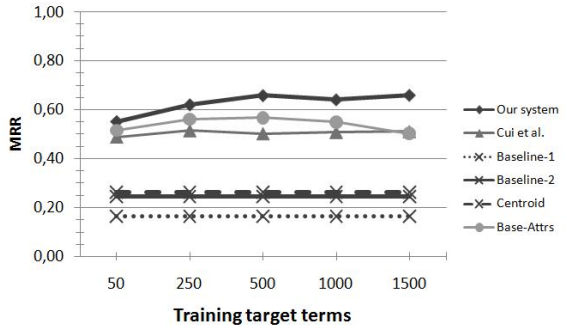


Figure 4: MRR scores, 5 answers per question.

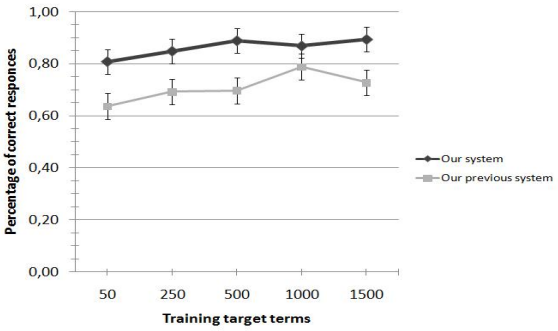


Figure 5: Correct responses of our new and previous system, allowing 5 answers per question.

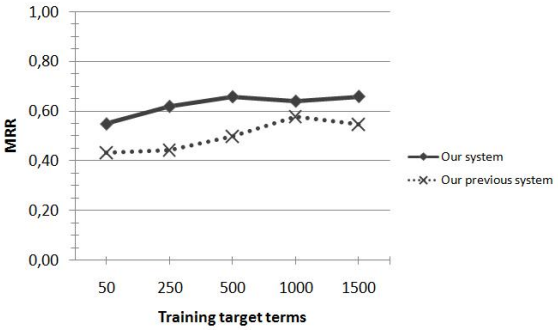


Figure 6: MRR of our new and previous system.

latter uses more linguistic resources (a POS tagger and a chunker). Both systems outperform the baselines, of which the centroid baseline is the best, and both systems perform better as the size of the training set increases. The baselines contain no learning components; hence, their curves are flat. We also show the results (Base-Attrs) of our system when the features that correspond to automatically acquired patterns are excluded. Clearly, these patterns help our system achieve significantly better results; however, our system outperforms Cui et al.'s even without them. Without the automatic patterns, our system also shows signs of saturation as the training data increase.

Figures 5 and 6 show the performance of our new system against our previously published one (Androutsopoulos and Galanis, 2005); the new system clearly outperforms the old one. Additional experiments we conducted with the old system replacing the SVM by the MAXENT classifier (without using ROUGE-W) indicate that the use of MAXENT by itself also improved slightly the results, but the differences are too minor to show; the improvement is mostly due to the use of ROUGE-W instead of our previous measure.

5 Related work

Xu et al. (2004) use an information extraction engine to extract linguistic features from documents relevant to the target term. The features are mostly phrases, such as appositives, and phrases expressing relations. The features are then ranked by their type and similarity to a centroid, and the most highly ranked ones are returned. Xu et al. seem to aim at generating multi-snippet definitions, unlike the single-snippet definitions we seek.

Blair-Goldensohn et al. (2003; 2004) extract sentences that may provide definitional information from documents retrieved for the target term; a decision tree learner and manually tagged training data are used. The sentences are then matched against manually constructed patterns, which operate on syntax trees, to detect sentences expressing the target term's genus, species, or both (genus+species). The system composes its answer by placing first the genus+species sentence that is closer to the centroid of the extracted sentences. The remaining sentences are ranked by their distance from the centroid, and the most highly ranked ones are clustered. The system then selects iteratively the cluster that is closer to the centroid of the extracted sentences and the most recently used cluster. The cluster's most representative sentence, i.e., the sentence closest to the centroid of the cluster's sentences, is added to the response. The iterations stop when a maximum response length is reached. Multi-snippet definitions are generated.

Han et al. (2004; 2006) parse a definition question to locate the head word of the target term. They also use a named entity recognizer to determine the target term's type (person, organization,

etc.). They then extract from documents relevant to the target term sentences containing its head word, as well as sentences the extracted ones refer to (e.g., via pronouns). The resulting sentences are matched against manually constructed syntactic patterns to detect phrases conveying definitional information. The resulting phrases are ranked by criteria like the degree to which the phrase contains words common in definitions of the target term's type, and the highest ranked phrases are included in a multi-snippet summary. Other mechanisms discard phrases duplicating information.

Xu et al. (2005) aim to extract all the definitions in a document collection. They parse the documents to detect base noun phrases (without embedded noun phrases). Base noun phrases are possible target terms; the paragraphs containing them are matched against manually constructed patterns that look for definitions. An SVM then separates the remaining paragraphs into good, indifferent, and bad definitions. Redundant paragraphs, identified by edit distance similarity, are removed.

6 Conclusions and future work

We presented a freely available system that finds short definitions of user-specified terms on Web pages. It employs a MAXENT classifier, which is trained on automatically generated examples; hence, the system is in effect unsupervised. We use ROUGE-W to generate training examples from Web snippets and encyclopedias, a method that outperforms an alternative centroid-based one. Once our system has been trained, it can find short definitions of terms that are not covered by encyclopedias. Experiments show our system outperforms a comparable well-published system and a previously published form of our system.

Our system does not require linguistic processing tools, such as named entity recognizers, POS taggers, chunkers, parsers; hence, it can be easily used in languages where such tools are unavailable. It could be improved by exploiting the HTML markup of Web pages and the Web's hyperlinks. For example, the target term is sometimes written in italics in definitions, and some definitions are provided on pages (e.g., pop-up windows) that occurrences of the target term link to.

The work reported here was conducted in the context of project INDIGO, where an autonomous robotic guide for museum collections is being developed (Galanis et al., 2009). The guide engages the museum's visitors in spoken dialogues, and it describes the exhibits the visitors select by generating spoken natural language descriptions from an ontology. Among other requests, the visitors can ask follow up questions, and we have found that the most common kind of follow up questions are requests to define terms (e.g., names of persons, events, architectural terms, etc.) mentioned in the generated exhibit descriptions. Some of these definition requests can be handled by generating new texts from the ontology, but some times the ontology contains no information for the target terms. We are, thus, experimenting with the possibility of obtaining short definitions from the Web, using the system we presented.

Acknowledgements

This work was carried out in INDIGO, an FP6 IST project funded by the European Union, with additional funding provided by the Greek General Secretariat of Research and Technology.[13]

References

Androutsopoulos, I., and Galanis, D. 2005. *A Practically Unsupervised Learning Method to Identify Single-Snippet Answers to Definition Questions on the Web*. In HLT/EMNLP, Vancouver, Canada, 323–330.

Blair-Goldensohn, S., McKeown, K., Schlaikjer, A.H. 2003. *A Hybrid Approach for QA Track Definitional Questions*. In TREC 2003, Gaithersburg, MD, USA.

Blair-Goldensohn, S., McKeown, K.R., and Schlaikjer, A.H. 2004. *Answering Definitional Questions: A Hybrid Approach*. In Maybury, M. (Ed.), New Directions in Question answering, AAAI Press.

Chu-Carroll, J., Czuba, K., Prager, J., Ittycheriah, A., Blair-Goldensohn, S. 2004. *IBM's PIQUANT II in TREC 2004*. In TREC 2004, Gaithersburg, MD, USA.

Chu-Carroll, J., Czuba, K., Duboue, P., and Prager, J. 2005. *IBM's PIQUANT II in TREC 2005*. In TREC 2005, Gaithersburg, MD, USA.

Cristianini, N. and Shawe-Taylor, J. 2000. *An Introduction to SVMs*. Cambridge University Press.

Cui, H., Kan, M.-Y., Chua, T.-S., and Xiao, J. 2004. *A Comparative Study on Sentence Retrieval for Definitional Question Answering*. In SIGIR workshop on Information Retrieval for Question Answering, Salvador, Brazil.

[13] Consult http://www.ics.forth.gr/indigo/.

Cui, H., Kan, M.Y., Chua, T.S. 2004. *Unsupervised Learning of Soft Patterns for Generating Definitions from Online News*. In WWW, New York, NY, USA.

Cui, H., Kan, M.Y., Chua, T.S. 2005. *Generic Soft Pattern Models for Definitional Question Answering*. In ACM SIGIR, Salvador, Brazil.

Cui, H. 2006. *Soft Matching for Question Answering*. Ph.D. thesis, National University of Singapore.

Cui, H., Kan, M., and Chua, T. 2007. *Soft Pattern Matching Models for Definitional Question Answering*. ACM Transactions on Information Systems, 25(2):1–30.

Dang, H. T. 2005. *Overview of DUC 2005*. In DUC at HLT-EMNLP, Vancouver, Canada.

Dang, H. T. 2006. *Overview of DUC 2006*. In DUC at HLT-NAACL, New York, NY, USA.

Eugenio, B. D., Glass, M. 2004. *The Kappa Statistic: a Second Look*. Computational Linguistics, 301(1):95-101.

Galanis, D., Karakatsiotis, G., Lampouras, G., and Androutsopoulos, I. 2009. *An Open-Source Natural Language Generator for* OWL *Ontologies and its Use in Protege and Second Life*. EACL system demonstration, Athens, Greece.

Han, K.S., Chung, H., Kim, S.B., Song, Y.I., Lee, J.Y., Rim, H.C. 2004. *Korea University QA System at TREC 2004*. In TREC 2004, Gaithersburg, MD, USA.

Han, K.S., Song, Y.I., Kim, S.B., and Rim, H.C. 2006. *A Definitional Question Answering System Based on Phrase Extraction Using Syntactic Patterns*. IEICE Transactions on Information and Systems, vol. E89-D, No. 4, 1601–1605.

Hildebrandt, W., Katz, B., and Lin, J. 2004. *Answering Definition Questions Using Multiple Knowledge Sources*. In HLT-NAACL, Boston, MA, USA, 49–56.

Joho, H. and Sanderson, M. 2000. *Retrieving Descriptive Phrases from Large Amounts of Free Text*. International Conference on Information and Knowledge Management, McLean, VA, USA, 180–186.

Joho, H. and Sanderson, M. 2001. *Large Scale Testing of a Descriptive Phrase Finder*. In HLT-NAACL, San Diego, CA, USA, 219–221.

Lin, C.Y. 2004. *ROUGE: A Package for Automatic Evaluation of Summaries*. In ACL workshop "Text Summarization Branches Out", Barcelona, Spain.

Miliaraki, S. and Androutsopoulos, I. 2004. *Learning to Identify Single-Snippet Answers to Definition Questions*. In COLING, Geneva, Switzerland, 1360–1366.

Prager, J., Radev, D., and Czuba, K. 2001. *Answering What-Is Questions by Virtual Annotation*. In HLT-NAACL, San Diego, CA, USA, 26–30.

Prager, J., Chu-Carroll, J., and Czuba, K. 2002. *Use of WordNet Hypernyms for Answering What-Is Questions*. In TREC 2001, Gaithersburg, MD, USA.

Ratnaparkhi A. 1997. *A Simple Introduction to Maximum Entropy Models for Natural Language Processing*. Technical Report 97-08, Institute for Research in Cognitive Science, University of Pennsylvania, 1997.

Voorhees, E.M. 2000. *Overview of the TREC-9 Question Answering Track*. NIST, USA.

Voorhees, E.M. 2001. *Overview of the TREC 2001 Question Answering Track*. NIST, USA.

Voorhees, E.M. 2001. *The TREC QA Track*. Natural Language Engineering, 7(4):361–378.

Voorhees, E.M. 2003. *Evaluating Answers to Definition Questions*. In HLT-NAACL, Edmonton, Canada.

Xu, J., Weischedel, R., Licuanan, A. 2004. *Evaluation of an Extraction-based Approach to Answering Definitional Questions*. In ACM SIGIR, Sheffield, UK.

Xu, J., Cao, Y., Li, H., Zhao, M. 2005. *Ranking Definitions with Supervised Learning Methods*. In WWW, Chiba, Japan, 811–819.

Improving Nominal SRL in Chinese Language with Verbal SRL Information and Automatic Predicate Recognition

Junhui Li[†] **Guodong Zhou**[†*] **Hai Zhao**[†‡] **Qiaoming Zhu**[†] **Peide Qian**[†]

[†] Jiangsu Provincial Key Lab for Computer Information Processing Technologies
School of Computer Science and Technology
Soochow University, Suzhou, China 215006

[‡] Department of Chinese, Translation and Linguistics
City University of HongKong, China

Email: {lijunhui,gdzhou,hzhao,qmzhu,pdqian}@suda.edu.cn

Abstract

This paper explores Chinese semantic role labeling (SRL) for nominal predicates. Besides those widely used features in verbal SRL, various nominal SRL-specific features are first included. Then, we improve the performance of nominal SRL by integrating useful features derived from a state-of-the-art verbal SRL system. Finally, we address the issue of automatic predicate recognition, which is essential for a nominal SRL system. Evaluation on Chinese NomBank shows that our research in integrating various features derived from verbal SRL significantly improves the performance. It also shows that our nominal SRL system much outperforms the state-of-the-art ones.

1. Introduction

Semantic parsing maps a natural language sentence into a formal representation of its meaning. Due to the difficulty in deep semantic parsing, most of previous work focuses on shallow semantic parsing, which assigns a simple structure (such as WHO did WHAT to WHOM, WHEN, WHERE, WHY, HOW) to each predicate in a sentence. In particular, the well-defined semantic role labeling (SRL) task has been drawing more and more attention in recent years due to its importance in deep NLP applications, such as question answering (Narayanan and Harabagiu, 2004), information extraction (Surdeanu et al., 2003), and co-reference resolution (Ponzetto and Strube, 2006). Given a sentence and a predicate (either a verb or a noun) in it, SRL recognizes and maps all the constituents in the sentence into their corresponding semantic arguments (roles) of the predicate. According to the predicate types, SRL could be divided into SRL for verbal predicates (verbal SRL, in short) and SRL for nominal predicates (nominal SRL, in short).

During the past few years, verbal SRL has dominated the research on SRL with the availability of FrameNet (Baker et al., 1998), PropBank (Palmer et al., 2005), and the consecutive CoNLL shared tasks (Carreras and Màrquez, 2004 & 2005) in English language. As a complement to PropBank on verbal predicates, NomBank (Meyers et al., 2004) annotates nominal predicates and their corresponding semantic roles using similar semantic framework as PropBank. As a representative, Jiang and Ng (2006) pioneered the exploration of various nominal SRL-specific features besides the traditional verbal SRL-related features on NomBank. They achieved the performance of 72.73 and 69.14 in F1-measure on golden and automatic syntactic parse trees, respectively, given golden nominal predicates.

For SRL in Chinese, Sun and Jurafsky (2004) and Pradhan et al. (2004) pioneered the research on Chinese verbal and nominal SRLs, respectively, on small private datasets. Taking the advantage of recent release of Chinese PropBank (Xue and Palmer, 2003) and Chinese NomBank (Xue, 2006a), Xue and his colleagues (Xue and Palmer 2005; Xue 2006b; Xue, 2008) pioneered the exploration of Chinese verbal and nominal SRLs, given golden predicates. Among them, Xue and Palmer (2005) studied Chinese verbal SRL using Chinese PropBank and achieved the performance of 91.3 and 61.3 in F1-measure on golden and automatic syntactic parse trees, respectively. Xue (2006b) extended their study on Chinese nominal SRL and attempted to improve the performance of nominal SRL by simply in-

[*] Corresponding author

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1280–1288,
Singapore, 6-7 August 2009.

cluding the Chinese PropBank training instances into the training data for nominal SRL on Chinese NomBank. However, such integration was empirically proven unsuccessful due to the different nature of certain features for verbal and nominal SRLs. Xue (2008) further improved the performance on both verbal and nominal SRLs with a better syntactic parser and new features. Ding and Chang (2008) focused on argument classification for Chinese verbal predicates with hierarchical feature selection strategy. They achieved the classification precision of 94.68% on golden parse trees on Chinese PropBank.

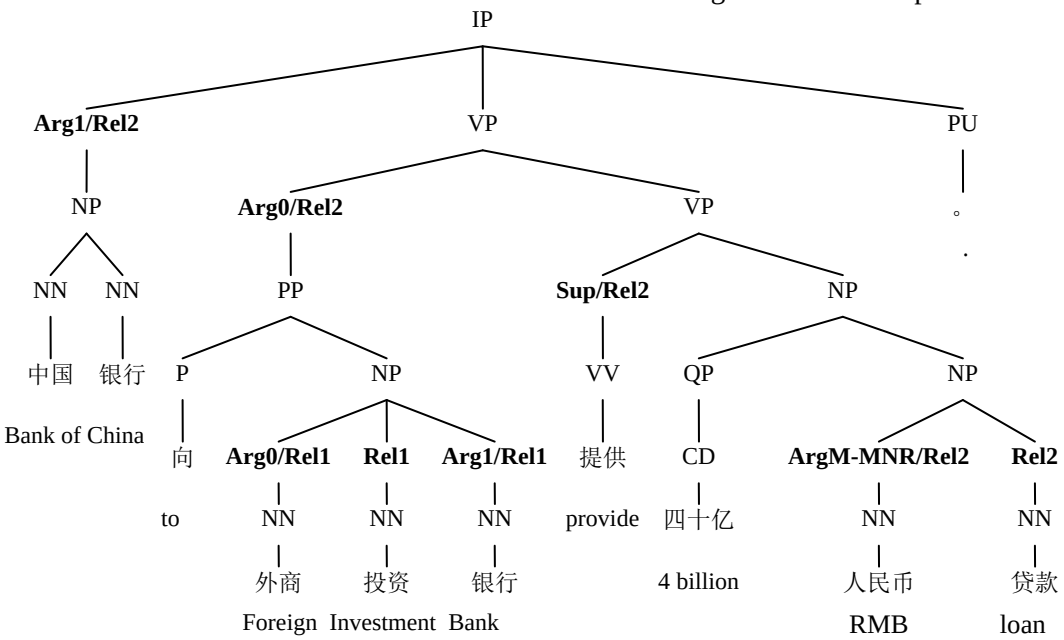
This paper focuses on Chinese nominal SRL. This is done by adopting a traditional verbal SRL architecture to handle Chinese nominal predicates with additional nominal SRL-specific features. Moreover, we significantly enhance the performance of nominal SRL by properly integrating various features derived from verbal SRL. Finally, this paper investigates the effect of automatic nominal predicate recognition on the performance of Chinese nominal SRL. Although previous research (e.g. CoNLL'2008) in English nominal SRL reveals the importance of automatic predicate recognition, there has no reported research on automatic predicate recognition in Chinese nominal SRL.

The rest of this paper is organized as follows: Section 2 introduces Chinese NomBank while the baseline nominal SRL system is described in Section 3 with traditional and nominal SRL-specific features. Then, the baseline nominal SRL system is improved by integrating useful features derived from verbal SRL (Section 4) and extended with automatic recognition of nominal predicates (Section 5). Section 6 gives experimental results and discussion. Finally, Section 7 concludes the paper.

2. Chinese NomBank

Chinese NomBank (Xue, 2006a) adopts similar semantic framework as NomBank, and focuses on Chinese nominal predicates with their arguments in Chinese TreeBank. The semantic arguments include:

1) Core arguments: Arg0 to Arg5. Generally, Arg0 and Arg1 denotes the agent and the patient, respectively, while arguments from Arg2 to Arg5 are predicate-specific.
2) Adjunct arguments, which are universal to all predicates, e.g. ArgM-LOC for locative, and ArgM-TMP for temporal.



Bank of China provides 4 billion RMB loan to Foreign Investment Bank.

Figure 1: Two nominal predicates and their arguments in the style of NomBank.

All the arguments are annotated on parse tree nodes with their boundaries aligning with the spans of tree nodes. Figure 1 gives an example with two nominal predicates and their respective arguments, while the nominal predicate "投资/investment" has two core arguments, "NN(外商/foreign businessman)" as Arg0 and "NN(银行/bank)" as Arg1, and the other nominal predicate "贷款/loan" also has two core arguments, "NP(中国银行/Bank of China)" as Arg1 and

"PP(向外商投资银行/to Foreign Investment Bank)" as Arg0, and 1 adjunct argument, "NN(人民币/RMB)" as ArgM-MNR, denoting the manner of loan. It is worth noticing that there is a (Chinese) NomBank-specific label in Figure 1, Sup (support verb) (Xue, 2006a), in helping introduce the arguments, which occur outside the nominal predicate-headed noun phrase. This is illustrated by the nominal predicate "贷款/loan", whose Arg0 and Arg1 are both realized outside the nominal predicate-headed noun phrase, NP(四十亿人民币贷款/4 billion RMB loan). Normally, a verb is marked as a support verb only when it shares some arguments with the nominal predicate.

3. Baseline: Chinese Nominal SRL

Popular SRL systems usually formulate SRL as a classification problem, which annotates each constituent in a parse tree with a semantic role label or with the non-argument label NULL. Besides, we divide the system into three consecutive phases so as to overcome the imbalance between the training instances of the NULL class and those of any other argument classes.

Argument pruning. Here, several heuristic rules are adopted to filter out constituents, which are most likely non-arguments. According to the argument structures of nominal predicates, we categorize arguments into two types: arguments inside NP (called ***inside arguments***) and arguments introduced via a support verb (called ***outside arguments***), and handle them separately. For the inside arguments, the following three heuristic rules are applied to find *inside argument candidates*:

- All the sisters of the predicate are candidates.
- If a CP or DNP node is a candidate, its children are candidates too.
- For any node X, if its parent is an ancestral node of the predicate, and the internal nodes along the path between X and the predicate are all NPs, then X is a candidate.

For outside arguments, we look for the support verb of the focus nominal predicate, and then adopt the rules as proposed in Xue and Palmer (2005) to find the candidates for the support verb, since *outside argument candidates* are introduced via this support verb. That to say, the argument candidates of the support verb are regarded as *outside argument candidates* of the nominal predicate. However, as support verbs are not annotated explicitly in the testing phase, we identify intervening verbs as alternatives to support verbs in both training and testing phases with the path between the nominal predicate and intervening verb in the form of "VV<VP>[NP>]$^{+}$NN", where "[NP>]$^{+}$" denotes one or more NPs. Our statistics on Chinese NomBank shows that 51.96% of nominal predicates have no intervening verb while 48.04% of nominal predicates have only one intervening verb.

Taken the nominal predicate "贷款/loan" in Figure 1 as an example, NN(人民币/RMB) and QP(四十亿/4 billion) are identified as *inside argument candidates*, while PP(向外商投资银行/to Foreign Investment Bank) and NP(中国银行/Bank of China) are identified as *outside argument candidates* via the support verb VV(提供/provide).

Argument identification. A binary classifier is applied to determine the candidates as either valid arguments or non-arguments. It is worth pointing out that we only mark those candidates that are most likely to be NULL (with probability > 0.90) as non-arguments. Our empirical study shows that this little trick much benefits nominal SRL, since argument identification for nominal predicates is much more difficult than that for verbal predicates and thus many arguments would have been falsely marked as non-arguments if the threshold is set as 0.5.

Argument classification. A multi-class classifier is employed to label identified arguments with specific argument labels (including the NULL class for non-argument).

In the following, we first adapt some traditional features, which have been proven effective in verbal SRL, to nominal SRL, and then introduce several nominal SRL-specific features.

3.1. Traditional Features

Using the feature naming convention as adopted in Jiang and Ng (2006), Table 1 lists the traditional features, where "I" and "C" indicate the features for argument identification and classification, respectively. Among them, the predicate class (b2) feature was first introduced in Xue and Palmer (2005) to overcome the imbalance of the predicate distribution in that some predicates can be only found in the training data while some predicates in the testing data are absent from the training data. In particular, the verb class is classified along three dimensions: the number of arguments, the number of framesets and selected syntactic alternations. For example,

the verb class of "C1C2a" means that it has two framesets, with the first frameset having one argument and the second having two arguments. The symbol "a" in the second frameset represents a type of syntactic alternation.

Feature	Remarks: b1-b5(C, I), b6-b7(C)
b1	Predicate: the nominal predicate itself. (贷款/loan)
b2	Predicate class: the verb class that the predicate belongs to. (C4a)
b3	Head word (b3H) and its POS (b3P). (银行/bank, NN)
b4	Phrase type: the syntactic category of the constituent. (NP)
b5	Path: the path from the constituent to the nominal predicate. (NP<IP>VP>VP>NP>NP>NN)
b6	Position: the positional relationship of the constituent with the predicate. "left" or "right". (left)
b7	First word (b7F) and last word (b7L) of the focus constituent. (中国/China, 银行/bank)
Combined features: b11-b14(C, I), b15(C)	
b11: b1&b4; b12: b1&b3H; b13: b2&b4; b14: b2&b3H; b15: b5&b6	

Table 1: Traditional features and their instantiations for argument identification and classification, with NP(中国银行/Bank of China) as the focus constituent and NN(贷款/loan) as the nominal predicate, regarding Figure 1.

3.2. Nominal SRL-specific Features

To capture more useful information in the predicate-argument structure, we also study additional features which provide extra information. Statistics on Chinese NomBank show that about 40% of pruned inside candidates are arguments. Since inside arguments usually locate near to the nominal predicate, its surroundings are expected to be helpful in SRL. Table 2 shows the features in better capturing the details between inside arguments and nominal predicates. Specially, features ai6 and ai7 are sister-related features, inspired by the features related with the neighboring arguments in Jiang and Ng (2006).

Statistics on NomBank and Chinese NomBank show that about 20% and 22% of arguments are introduced via a support verb, respectively. Since a support verb pivots outside arguments and the nominal predicate on its two sides, support verbs play an important role in labeling these arguments. Here, we also identify intervening verbs as alternatives to support verbs since support verbs are not explicitly in the testing phase. Table 3 lists the intervening verb-related features (ao1-ao4, ao11-ao14) employed in this paper.

Feature	Remarks
ai1	Whether the focus constituent is adjacent to the predicate. Yes or No. (Yes)
ai2	The headword (ai2H) and pos (ai2P) of the predicate's nearest right sister. (银行/bank, NN)
ai3	Whether the predicate has right sisters. Yes or No. (Yes)
ai4	Compressed path of b5: compressing sequences of identical labels into one. (NN<NP>NN)
ai5	Whether the predicate has sisters. Yes or No. (Yes)
ai6	For each sister of the focus constituent, combine b3H&b4&b5&b6. (银行/bank&NN & NN<NP>NN&right)
ai7	Coarse version of ai6, b4&b6. (NN&right)

Table 2: Additional features and their instantiations for inside argument candidates, with "NN(外商/foreign businessman)" as the focus constituent and "NN(投资/investment)" as the nominal predicate, regarding Figure1.

Feature	Remarks
ao1	Intervening verb itself. (提供/provide)
ao2	The verb class that the intervening verb belongs to. (C3b)
ao3	The path from the focus constituent to the intervening verb. (NP<IP>VP>VP>VV)
ao4	The compressed path of ao3: compressing sequences of identical labels into one. (NP<IP>VP>VV)
Combined features: ao11-ao14	
ao11: ao1&ao3; ao12: ao1&ao4; ao13: ao2&ao3; ao14: ao2&ao4.	

Table 3: Additional features and their instantiations for outside argument candidates, with "NP(中国银行/Bank of China)" as the focus constituent and "贷款/loan" as the nominal predicate, regarding Figure1.

Feature selection. Some Features proposed above may not be effective in tasks of identification and classification. We adopt the greedy feature selection algorithm as described in Jiang and Ng (2006) to pick up positive features empirically and incrementally according to their contributions on the development data. The algorithm repeatedly selects one feature each time which contributes most, and stops when adding any of the remaining features fails to improve the performance. As far as the SRL task concerned, the whole feature selection process could be done as follows: 1). Feature selection for argument identification: run the selection algo-

rithm with the basic set of features (b1-b5, b11-b14) to pick up effective features from (ai1-ai7, ao1-ao4, ao11-ao14); 2). Feature selection for argument classification: fix the output returned in step1 as the feature set of argument identification, and run the selection algorithm with the basic set of features (b1-b7, b11-b15) to select positive features from (ai1-ai7, ao1-ao4, ao11-ao14) for argument classification.

4. Integrating Features derived from Verbal SRL

Since Chinese PropBank and NomBank are annotated on the same data set with the same lexical guidelines (e.g. frame files), it may be interesting to investigate the contribution of Chinese verbal SRL on the performance of Chinese nominal SRL. In the frame files, argument labels are defined with regard to their semantic roles to the predicate, either a verbal or nominal predicate. For example, in the frame file of predicate "贷款/loan", the borrower is always labeled with Arg0 and the lender labeled with Arg1. This can be demonstrated by the following two sentences: "贷款/loan" is annotated as a nominal and a verbal predicate in S1 and S2, respectively.

S1 [Arg1 中国银行/Bank of China] [Arg0 向外商投资银行/to Foreign Investment Bank] 提供/provide [Rel 贷款/loan]

S2 [Arg0 中国银行/Bank of China] [Arg1 向外商投资银行/from Foreign Investment Bank] [Rel 贷款/loan]

Therefore, it is straightforward to augment nominal training instances with verbal ones. However, Xue (2006b) found that simply adding the training instances for verbal SRL to the training data for nominal SRL and indiscriminately extracting the same features in both verbal and nominal SRLs hurt the performance. This may be due to that certain features (e.g. the path feature) are much different for verbal and nominal SRLs. This can be illustrated in sentences S1 and S2: the verbal instances in S2 are negative for semantic role labeling of the nominal predicate "贷款/loan" in S1, since "中国银行/Bank of China" takes opposite roles in S1 and S2. So does "向外商投资银行/(from/to) Foreign Investment Bank".

Although several support verb-related features (ao1-ao4, ao11-ao14) have been proposed, one may still ask how large the role support verbs can play in nominal SRL. It is interesting to note that outside arguments and the highest NP phrase headed by the nominal predicate are also annotated as arguments of the support verb in Chinese PropBank. For example, Chinese PropBank marks "中国银行/Bank of China" as Arg0 and "四十亿人民币贷款/4 billion RMB loan" as Arg1 for verb "提供/provide" in Figure1. Let OA be the outside argument, VV be the support verb, and NP be the highest NP phrase headed by the nominal predicate NN, then there exists a pattern "OA VV NN" in the sentence, where the support verb VV plays a certain role in transferring roles between OA and NN. For example, if OA is the agent of VV, then OA is also the agent of phrase VP(VV NN). Like the example in Figure1, supposing a NP is the agent of support verb "提供/provide" as well as VP phrase ("提供四十亿人民币贷款/provide 4 billion RMB loan"), we can infer that the NP is the lender of the nominal predicate "贷款/loan" independently on any other information, such as the NP content and the path from the NP to the nominal predicate "贷款/loan".

Let C be the focus constituent, V be the intervening verb, and NP be the highest NP headed by the nominal predicate. Table 4 shows the features (ao5-ao8, p1-p7) derived from verbal SRL. In this paper, we develop a state-of-the-art Chinese verbal SRL system, similar to the one as shown in Xue (2008), to achieve the goal. Based on golden parse trees on Chinese PropBank, our Chinese verbal SRL system achieves the performance of 92.38 in F1-measure, comparable to Xue (2008) which achieved the performance of 92.0 in F1-measure.

Feature	Remarks
ao5	Whether C is an argument for V. Yes or No
ao6	The semantic role of C for V.
ao7	Whether NP is an argument for V. Yes or No
ao8	The semantic role of NP for V.
Combined features: p1-p7	
p1: ao1&ao5; p2: ao1&ao6; p3: ao1&ao5&b1; p4: ao1&ao6&b1; p5: ao1&apo7; p6: ao1&ao8; p7: ao5&ao7.	

Table 4: Features derived from verbal SRL.

5. Automatic Predicate Recognition

Unlike Chinese PropBank where almost all the verbs are annotated as predicates, Chinese NomBank only marks those nouns having arguments as predicates. Statistics on Chinese NomBank show that only 17.5% of nouns are marked as predicates. It is possible that a noun is a predi-

cate in some cases but not in others. Previous Chinese nominal SRL systems (Xue, 2006b; Xue, 2008) assume that nominal predicates have already been manually annotated and thus are available. To our best knowledge, there is no report on addressing automatic recognition of nominal predicates on Chinese nominal SRL.

Automatic recognition of nominal predicates can be cast as a binary classification (e.g., Predicate vs. Non-Predicate) problem. This paper employs the convolution tree kernel, as proposed in Collins and Duffy (2001), on automatic recognition of nominal predicates.

Given the convolution tree kernel, the key problem is how to extract a parse tree structure from the parse tree for a nominal predicate candidate. In this paper, the parse tree structure is constructed as follows: 1) starting from the predicate candidate's POS node, collect all of its sister nodes (with their headwords); 2). recursively move one level up and collect all of its sister nodes (with their headwords) till reaching a non-NP node. Specially, in order to explicitly mark the positional relation between a node and the predicate candidate, all nodes on the left side of the candidate are augmented with tags 1 and 2 for nodes on the right side. Figure 2 shows an example of the parse tree structure with regard to the predicate candidate "贷款/loan" as shown in Figure 1.

In our extra experiments we found global statistic features (e.g. g1-g5) about the predicate candidate are helpful in a feature vector-based method for predicate recognition. Figure 2 makes an attempt to utilize those features in kernel-based method. We have explored other ways to include those global features. However, the way in Figure 2 works best.

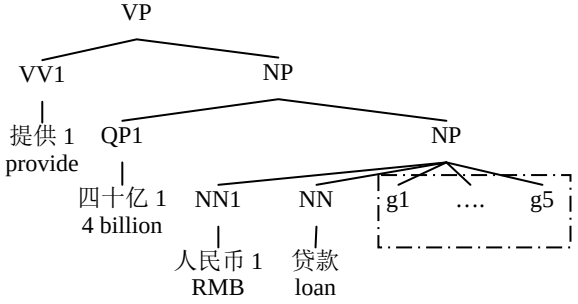


Figure 2: Semantic sub-tree for nominal predicate

Let the predicate candidate be w_0, and its left and right neighbor words be w_{-1} and w_1, respectively. The five global features are defined as follows.

g1 Whether w_0 is ever tagged as a verb in the training data? Yes or No.

g2 Whether w_0 is ever annotated as a nominal predicate in the training data? Yes or No.

g3 The most likely label for w_0 when it occurs together with w_{-1} and w_1.

g4 The most likely label for w_0 when it occurs together with w_{-1}.

g5 The most likely label for w_0 when it occurs together with w_1.

6. Experiment Results and Discussion

We have evaluated our Chinese nominal SRL system on Chinese NomBank with Chinese PropBank 2.0 as its counterpart.

6.1. Experimental Settings

This version of Chinese NomBank consists of standoff annotations on the files (chtb_001 to 1151.fid) of Chinese Penn TreeBank 5.1. Following the experimental setting in Xue (2008), 648 files (chtb_081 to 899.fid) are selected as the training data, 72 files (chtb_001 to 040.fid and chtb_900 to 931.fid) are held out as the test data, and 40 files (chtb_041 to 080.fid) as the development data, with 8642, 1124, and 731 propositions, respectively.

As Chinese words are not naturally segmented in raw sentences, two Chinese automatic parsers are constructed: word-based parser (assuming golden word segmentation) and character-based parser (with automatic word segmentation). Here, Berkeley parser (Petrov and Klein, 2007)[1] is chosen as the Chinese automatic parser. With regard to character-based parsing, we employ a Chinese word segmenter, similar to Ng and Low (2004), to obtain the best automatic segmentation result for a given sentence, which is then fed into Berkeley parser for further syntactic parsing. Both the word segmenter and Berkeley parser are developed with the same training and development datasets as our SRL experiments. The word segmenter achieves the performance of 96.1 in F1-measure while the Berkeley parser gives a performance of 82.5 and 85.5 in F1-measure on golden and automatic word segmentation, respectively[2].

In addition, SVMLight with the tree kernel function (Moschitti, 2004)[3] is selected as our classifier. In order to handle multi-classification

[1] Berkeley Parser. http://code.google.com/p/berkeleyparser/

[2] POSs are not counted in evaluating the performance of word-based syntactic parser, but they are counted in evaluating the performance of character-based parser. Therefore the F1-measure for the later is higher than that for the former.

[3] SVM-LIGHT-TK. http://dit.unitn.it/~moschitt/

problem in argument classification, we apply the *one vs. others* strategy, which builds K classifiers so as to separate one class from all others. For argument identification and classification, we adopt the linear kernel and the training parameter C is fine-tuned to 0.220. For automatic recognition of nominal predicates, the training parameter C and the decay factor λ in the convolution tree kernel are fine-tuned to 2.0 and 0.2, respectively.

6.2. Results with Golden Parse Trees and Golden Nominal Predicates

Effect of nominal SRL-specific features

	Rec.(%)	Pre.(%)	F1
traditional features	62.83	73.58	67.78
+nominal SRL-specific features	69.90	75.11	72.55

Table 5: The performance of nominal SRL on the development data with golden parse trees and golden nominal predicates

After performing the greedy feature selection algorithm on the development data, features {ao1, ai6, ai2P, ai5, ao2, ao12, ao14}, as proposed in Section 3.2, are selected consecutively for argument identification, while features {ai7, ao1, ai1, ao2, ai5, ao4} are selected for argument classification. Table 5 presents the SRL results on the development data. It shows that nominal SRL-specific features significantly improve the performance from 67.78 to 72.55 ($\chi^2; p < 0.05$) in F1-measure.

Effect of features derived from verbal SRL

Features	Rec.(%)	Pre.(%)	F1
baseline	67.86	73.63	70.63
+ao5	68.15	73.60	70.77 (+0.14)
+ao6	67.66	72.80	**70.14 (-0.49)**
+ao7	68.20	75.41	71.62 (+0.99)
+ao8	68.30	75.39	71.67 (+1.04)
+p1	67.91	74.40	71.00 (+0.37)
+p2	67.76	74.20	70.83 (+0.20)
+p3	67.96	74.69	71.16 (+0.53)
+p4	68.01	74.18	70.96 (+0.33)
+p5	68.01	75.01	71.39 (+0.76)
+p6	68.20	75.12	71.49 (+0.86)
+p7	68.40	75.70	71.87 (+1.24)

Table 6: Effect of features derived from verbal SRL on the performance of nominal SRL on the test data with golden parse trees and golden nominal predicates. The first row presents the performance using traditional and nominal SRL-specific features.

	Rec.(%)	Pre.(%)	F1
baseline	67.86	73.63	70.63
+features derived from verbal SRL	68.40	77.51	72.67
Xue (2008)	66.1	73.4	69.6

Table 7: The performance of nominal SRL on the test data with golden parse trees and golden nominal predicates

Table 6 shows the effect of features derived from verbal SRL in an incremental way. It shows that only the feature ao6 has negative effect due to its strong relevance with intervening verbs and thus not included thereafter. Table 7 shows the performance on the test data with or without using the features derived from the verbal SRL system. It shows these features significantly improve the performance ($\chi^2; p < 0.05$) on nominal SRL. Table 7 also shows our system outperforms Xue (2008) by 3.1 in F1-measure.

6.3. Results with Automatic Parse Trees and Golden Nominal Predicates

In previous section we have assumed the availability of golden parse trees during the testing process. Here we conduct experiments on automatic parse trees, using the Berkeley parser. Since arguments come from constituents in parse trees, those arguments, which do not align with any syntactic constituents, are simply discarded. Moreover, for any nominal predicate segmented incorrectly by the word segmenter, all its arguments are unable to be labeled neither. Table 8 presents the SRL performance on the test data by using automatic parse trees. It shows that the performance drops from 72.67 to 60.87 in F1-measure when replacing golden parse trees with word-based automatic ones, partly due to the absence of 6.9% arguments in automatic trees, and wrong POS tagging of nominal predicates. Table 8 also compares our system with Xue (2008). It shows that our system also outperforms Xue (2008) on Chinese NomBank.

	Rec. (%)	Pre. (%)	F1
This paper	56.95(53.55)	66.74(66.69)	60.87(59.40)
Xue (2008)	53.1 (52.9)	62.9 (62.3)	57.6 (57.3)

Table 8: The performance of nominal SRL on the test data with automatic parse trees and golden predicates. Here, the numbers outside the parentheses indicate the performance using a word-based parser, while the numbers inside indicate the performance using a character-based parser[4].

[4] About 1.6% nominal predicates are mistakenly segmented by the character-based parser, thus their arguments are missed directly.

6.4. Results with Automatic Nominal Predicates

So far nominal predicates are assumed to be manually annotated and available. Here we turn to a more realistic scenario in which both the parse tree and nominal predicates are automatically obtained. In the following, we first report the results of automatic nominal predicate recognition and then the results of nominal SRL on automatic recognition of nominal predicates.

Results of nominal predicate recognition

Parses	g1-g5	Rec.(%)	Pre.(%)	F1
golden	no	91.46	88.93	90.18
	yes	92.62	89.36	90.96
word-based	yes	86.39	81.80	84.03
character-based	yes	84.79	81.94	83.34

Table 9: The performance of automatic nominal predicate recognition on the test data

Table 9 lists the predicate recognition results, using the parse tree structure, as shown in Section 5, and the convolution tree kernel, as proposed in Collins and Duffy (2001). The second column (g1-g5) indicates whether the global features (g1-g5) are included in the parse tree structure. We have also defined a simple rule that treats a noun which is ever a verb or a nominal predicate in the training data as a nominal predicate. Based on golden parse trees, the rule receives the performance of 81.40 in F1-measure. This suggests that our method significantly outperforms the simple rule-based one. Table 9 also shows that:

- As a complement to local structural information, global features improve the performance of automatic nominal predicate recognition by 0.78 in F1-measure.
- The word-based syntactic parser decreases the F1-measure from 90.96 to 84.03, mostly due to the POSTagging errors between NN and VV, while the character-based syntactic parser further drops the F1-measure by 0.69, due to automatic word segmentation.

Results with automatic predicates

Parses	Predicates	Rec.(%)	Pre.(%)	F1
golden	golden	68.40	77.51	72.67
	automatic	65.07	74.65	69.53
word-based	golden	55.95	66.74	60.87
	automatic	52.67	59.56	55.90
character-based	golden	53.55	66.69	59.40
	automatic	50.66	59.60	54.77

Table 10: The performance of nominal SRL on the test data with the choices of golden/automatic parse trees and golden/automatic predicates

In order to have a clear performance comparison among nominal SRL on golden/automatic parse trees and golden/automatic predicates, Table 10 lists all the results in those scenarios.

6.5. Comparison

Chinese nominal SRL vs. Chinese verbal SRL

Comparison with Xue (2008) shows that the performance of Chinese nominal SRL is about 20 lower (e.g. 72.67 vs. 92.38 in F1-measure) than that of Chinese verbal SRL, partly due to the smaller amount of annotated data (about 1/5) in Chinese NomBank than that in Chinese PropBank. Moreover, according to Chinese NomBank annotation criteria (Xue 2006a), even when a noun is a true deverbal noun, not all of its modifiers are legitimate arguments or adjuncts of this predicate. Only arguments that can co-occur with both the nominal and verbal forms of the predicate are considered in the NomBank annotation. This means that the judgment of arguments is semantic rather than syntactic. These facts may also partly explain the lower nominal SRL performance, especially the performance of argument identification. This can be illustrated by the statistics on the development data that 96% (40%) of verbal (nominal) predicates' sisters are annotated as arguments. Finally, the predicate-argument structure of nominal predicates is more flexible and complicated than that of verbal predicates as illustrated in Xue (2006a).

Chinese nominal SRL vs. English nominal SRL

Liu and Ng (2007) reported the performance of 77.04 and 72.83 in F1-measure on English NomBank when golden and automatic parse trees are used, respectively. Taking into account that Chinese verbal SRL achieves comparable performance with English verbal SRL on golden parse trees, the performance gap between Chinese and English nominal SRL (e.g. 72.67 vs. 77.04 in F1-measure) presents great challenge for Chinese nominal SRL. Moreover, while automatic parse trees only decrease the performance of English nominal SRL by about 4.2 in F1-measure, automatic parse trees significantly decrease the performance of Chinese nominal SRL by more than 12 in F1-measure due to the much lower performance of Chinese syntactic parsing.

7. Conclusion

In this paper we investigate nominal SRL in Chinese language. In particular, some nominal SRL-specific features are included to improve

the performance. Moreover, various features derived from verbal SRL are properly integrated into nominal SRL. Finally, a convolution tree kernel is adopted to address the issue of automatic nominal predicates recognition, which is essential in a nominal SRL system.

To our best knowledge, this is the first research on

1) Exploring Chinese nominal SRL on automatic parse trees with automatic predicate recognition;
2) Successfully integrating features derived from Chinese verbal SRL into Chinese nominal SRL with much performance improvement.

Acknowledgement

This research was supported by Project 60673041 and 60873150 under the National Natural Science Foundation of China, Project 2006AA01Z147 under the "863" National High-Tech Research and Development of China, and Project BK2008160 under the Natural Science Foundation of the Jiangsu province of China. We also want to thank Dr. Nianwen Xue for share of the verb class file. We also want to thank the reviewers for insightful comments.

References

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

Xavier Carreras and Lluis Màrquez. 2004. Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL 2004.*

Xavier Carreras and Lluis Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL 2005*.

Michael Collins and Nigel Duffy. 2001. Convolution Kernels for Natural Language. In *Proceedings of NIPS 2001*.

Weiwei Ding and Baobao Chang. 2008. Improving Chinese Semantic Role Classification with Hierarchical Feature Selection Strategy. In *Proceedings of EMNLP 2008.*

Zheng Ping Jiang and Hwee Tou Ng. 2006. Semantic role Labeling of NomBank: a Maximum Entropy Approach. In *Proceedings of EMNLP 2006*.

Chang Liu and Hwee Tou Ng. 2007. Learning Predictive Structures for Semantic Role Labeling of NomBank. In *Proceedings of ACL 2007*.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Yong, and R. Grishman. 2004. Annotating Noun Argument Structure for NomBank. In *Proceedings of LREC 2004*.

Alessandro Moschitti. 2004. A Study on Convolution Kernels for Shallow Semantic Parsing. In *Proceedings of ACL 2004*.

Srini Narayanan and Sanda Harabagiu. 2004. Question Answering based on Semantic Structures. In *Proceedings of COLING 2004*.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese Part-of-speech Tagging: One-at-a-time or All-at-once? Word-based or Character-based? In *Proceedings of EMNLP 2004*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*.

Slav Petrov. and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proceesings of NAACL 2007*.

Simone Paolo Ponzetto and Michael Strube. 2006. Semantic Role Labeling for Coreference Resolution. In *Proceedings of EACL 2006*.

Sameer Pradhan, Honglin Sun, Wayne Ward, James H. Martin, and Dan Jurafsky. 2004. Parsing Arguments of Nominalizations in English and Chinese. In *Proceedings of NAACL-HLT 2004*.

Honglin Sun and Daniel Jurafsky. 2004. Shallow Semantic Parsing of Chinese. In *Proceedings of NAACL 2004*.

Mihai Surdeanu, Sanda Harabagiu, John Williams and Paul Aarseth. 2003. Using Predicate-argument Structures for Information Extraction. In *Proceedings of ACL 2003*.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluis Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of CoNLL 2008*.

Nianwen Xue and Martha Palmer. 2003. Annotating the Propositions in the Penn Chinese TreeBank. In *Proceedings of 2nd SIGHAN Workshop on Chinese Language Processing.*

Nianwen Xue and Martha Palmer. 2005. Automatic Semantic Role Labeling for Chinese verbs. In *Proceedings of IJCAI 2005*.

Nianwen Xue. 2006a. Annotating the Predicate-Argument Structure of Chinese Nominalizations. In *Proceedings of the LREC 2006*.

Nianwen Xue. 2006b. Semantic Role Labeling of Nominalized Predicates in Chinese. In *Proceedings of HLT-NAACL 2006*.

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

On the Use of Virtual Evidence in Conditional Random Fields

Xiao Li
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
xiaol@microsoft.com

Abstract

Virtual evidence (VE), first introduced by (Pearl, 1988), provides a convenient way of incorporating prior knowledge into Bayesian networks. This work generalizes the use of VE to undirected graphical models and, in particular, to conditional random fields (CRFs). We show that VE can be naturally encoded into a CRF model as potential functions. More importantly, we propose a novel semi-supervised machine learning objective for estimating a CRF model integrated with VE. The objective can be optimized using the Expectation-Maximization algorithm while maintaining the discriminative nature of CRFs. When evaluated on the *CLASSIFIEDS* data, our approach significantly outperforms the best known solutions reported on this task.

1 Introduction

Statistical approaches to sequential labeling problems rely on necessary training data to model the uncertainty of a sequence of events. Human's prior knowledge about the task, on the other hand, often requires minimum cognitive load to specify, and yet can provide information often complementary to that offered by a limited amount of training data. Whenever prior knowledge becomes available, it is desired that such information is integrated to a probabilistic model to improve learning.

Virtual evidence (VE), first introduced by Pearl (1988), offers a principled and convenient way of incorporating external knowledge into Bayesian networks. In contrast to *standard evidence* (also known as observed variables), VE expresses a prior belief over values of random variables. It has been shown that VE can significantly extend the modeling power of Bayesian networks without complicating the fundamental inference methodology (Bilmes, 2004; Reynolds and Bilmes, 2005).

This work extends the use of VE to undirected graphical models and, in particular, to conditional random fields (CRFs). We show that VE can be naturally encoded into an undirected graphical model as potential functions. More importantly, we discuss a semi-supervised machine learning setting for estimating CRFs with the presence of VE. As the conditional likelihood objective of CRFs is not directly maximizable with respect to unlabeled data, we propose a novel semi-supervised learning objective that can be optimized using the Expectation-Maximization (EM) algorithm while maintaining the discriminative nature of CRFs.

We apply our model to the CLASSIFIEDS data (Grenager et al., 2005). Specifically, we use VE to incorporate into a CRF model two types of prior knowledge specified in previous works. The first is defined based on the notion of *prototypes*, *i.e.*, example words for a given label; and the other assumes that adjacent tokens tend to have the same label. When unlabeled data becomes available, we further extend the sparse prototype information to other words based on distributional similarity. This results in so-called *collocation lists*, each consisting of a relatively large number of noisy "prototypes" for a label. Given the fact that these noisy prototypes are often located close to each other in an input sequence, we create a new type of VE based on word collocation to reduce ambiguity.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1289–1297,
Singapore, 6-7 August 2009.

We compare our CRF model integrated with VE with two state-of-the-art models, *i.e.*, constraint-driven learning (Chang et al., 2007) and generalized expectation criteria (Mann and McCallum, 2008). Experiments show that our approach leads to sequential labeling accuracies superior to the best results reported on this task in both supervised and semi-supervised learning.

2 Related work

There have been various works that make use of prior knowledge in sequential labeling tasks. Grenager et al. (2005) explicitly constrain the transition matrix of a hidden Markov model (HMM) to favor self transitions, assuming that fields tend to consist of consecutive runs of the same label.

Prototype-drive learning (Haghighi and Klein, 2006) specifies prior knowledge by providing a few *prototypes* (*i.e.*, canonical example words) for each label. This sparse prototype information is then propagated to other words based on distributional similarity. The relation between words and their prototypes are then used as features in a Markov random field (MRF) model. Since an MRF model aims to optimize the joint probability $p(\mathbf{x}, \mathbf{y})$ of input and state sequences, it is possible to apply the EM algorithm for unsupervised/semi-supervised learning.

Constraint-driven learning (Chang et al., 2007) expresses several kinds of constraints in a unified form. In inference, a new decision function is proposed to penalize the violation of the desired constraints as follows,

$$\underset{\mathbf{y}}{\text{argmax}}\ \lambda \cdot F(\mathbf{x}, \mathbf{y}) - \sum_k \rho_k d(\mathbf{y}, 1_{C_k}(\mathbf{x})) \quad (1)$$

Here $\lambda \cdot F(\mathbf{x}, \mathbf{y})$ is a linear decision function applicable to a number of sequential models, such as HMMs, MRFs and CRFs. Function d is implemented as the Hamming distance (or its approximation) between a hypothesis sequence and the space of state sequences that satisfy the constraint C_i. Due to the nature of the distance function, their work approximates EM training by finding the top K hypothesis sequences and using them as newly labeled instances to update the model. This process is repeated for a number of iterations in a self-training fashion (Yarowsky, 1995).

Generalized expectation criteria (Mann and McCallum, 2008) represent prior knowledge as *labeled features*, and use such information to regularize semi-supervised learning for CRFs. Formally, their learning objective consists of the standard CRF training objective, plus a Gaussian prior on model parameters and an additional regularization term:[1]

$$\sum_i \log p_\lambda(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) - \frac{1}{2\sigma^2}\|\lambda\|^2 - \rho D(\hat{p}||\tilde{p}_\lambda) \quad (2)$$

In the last term, $\hat{p}$ and $\tilde{p}_\lambda$ both refer to conditional distributions of labels given a feature. While the former is specified by prior knowledge, and the latter is estimated from unlabeled data.

Our approach incorporates prior knowledge as virtual evidence to express preferences over the values of a set of random variables. The notion of VE was first introduced by Pearl (1998) and further developed by Bilmes (2004), both in the context of Bayesian networks. Different from constraint-driven learning, VE can be formally encoded as part of a graphical model. The fundamental inference methodology, therefore, does not need to be altered. Moreover, VE has the flexibility of representing various kinds of prior knowledge. For example, Reynolds and Bilmes (2005) use VE that explicitly favors self transitions in dynamic Bayesian networks.

This work extends the use of VE to CRFs. In essence, VE herein can be viewed as probabilistic constraints in an undirected graph that allow exact inference. One of the biggest challenges of such a model lies in the semi-supervised machine learning setting. Since the entire state sequence of an unlabeled instance remains hidden, the conditional likelihood objective of CRFs is not directly optimizable. There have been a number of works that address this problem for conditional models. For example, *minimum entropy regularization* (Grandvalet and Bengio, 2004; Jiao et al., 2006), aims to maximize the conditional likelihood of labeled data while minimizing the conditional entropy of unlabeled data:

$$\sum_i \log p_\lambda(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) - \frac{1}{2\sigma^2}\|\lambda\|^2 - \rho H(\mathbf{y}|\mathbf{x}) \quad (3)$$

This approach generally would result in "sharper" models which can be data-sensitive in practice.

Another approach (Suzuki and Isozaki, 2008) embeds a joint probability model (HMM in their

[1] We slightly modify the notation here to be consistent with the rest of the paper.

case) into a CRF model as a new potential function. Semi-supervised learning is then conducted by iteratively (1) fixing the HMM and updating CRF parameters on labeled data and (2) fixing the CRF model and updating the HMM on unlabeled data.

Additionally, when unlabeled instances have partial labeling information, it is possible to optimize a marginal distribution of the conditional likelihood, *i.e.*, $p_\lambda(\mathbf{y}_o^{(i)}|\mathbf{x})$, on unlabeled data. Here $\mathbf{y}_o^{(i)}$ is a subvector of $\mathbf{y}^{(i)}$ that denotes the set of observed state variables. The optimization can be done in a similar fashion as training a hidden-state CRF model (Quattoni et al., 2007).

3 Task

We consider the problem of extracting fields from free-text advertisements. We use the CLASSIFIEDS data (Grenager et al., 2005) which consists of 8767 ads for apartment rental. 302 of the ads in the CLASSIFIEDS data have been manually-labeled with 12 fields, including *size*, *rent*, *neighborhood* and so on. The labeled data has been divided into train/dev/test sets with 102/100/100 ads respectively. The evaluation metric is the token-level accuracy where tokens include both words and punctuations.

Our goal in this work is two folds: (1) leverage both the training data and the prior knowledge specified for this task for supervised learning, and (2) additionally use the unlabeled data for semi-supervised learning. We exploit two types of prior knowledge:

- K1: *label consistency with prototypes*;
- K2: *label consistency within a sentence*.

K1 involves a set of prototype lists. Each list is attached with a label and consists of a set of example words for that label. In this work, we use the prototype lists originally defined by Haghighi and Klein (2006) (HK06) and subsequently used by Chang et al. (2005) (CRR07) and Mann and McCallum (2008) (MM08). The labels as well as their prototypes are shown in the first two columns of Table 1. Our model is desired to be consistent with such prototype information. Secondly, K2 means that tokens tend to have consistent labels within a sentence. A similar type of prior knowledge is implemented by CRR07 as a constraint in inference.

4 Conditional Random Fields

Conditional random fields are a probabilistic model that directly optimizes the conditional probability of a state (label) sequence given an input sequence (Lafferty et al., 2001). Formally, we let $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ denote an input sequence of T tokens, and $\mathbf{y} = (y_1, y_2, \ldots, y_T)$ the corresponding state sequence. We further augment $\mathbf{y}$ with two special states, *Start* and *End*,[2] represented by y_0 and y_{T+1} respectively. A linear-chain CRF model is an undirected graphical model as depicted in Figure 1(a), with the conditional probability given by

$$p_\lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\lambda(\mathbf{x})} \prod_t \psi_\lambda^{(t)}(\mathbf{x}, y_{t-1}, y_t) \qquad (4)$$

The partition function $Z_\lambda(\mathbf{x})$ normalizes the exponential form to be a probability distribution. $\psi_\lambda^{(t)}$ are a set of potential functions defined on the *maximum cliques* of the graph, *i.e.*, $(\mathbf{x}, y_{t-1}, y_t)$ in the case of a linear-chain CRF model. The potential functions are typically in the form of

$$\psi_\lambda^{(t)}(\mathbf{x}, y_{t-1}, y_t) = \exp\Big(\lambda \cdot f(\mathbf{x}, y_{t-1}, y_t, t)\Big) \qquad (5)$$

where λ is a weight vector and f is a feature vector of arbitrary functions of the corresponding clique.

Given a set of labeled examples $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^m$, we can estimate model parameters in a supervised machine learning setting. The objective is to estimate λ that maximizes the conditional likelihood while regularizing the model size:

$$\mathcal{L}_1 = \sum_{i=1}^m \log p_\lambda(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) - \frac{1}{2\sigma^2}\|\lambda\|^2 \qquad (6)$$

In this work, we optimize $\mathcal{L}_1$ using stochastic gradient descent and use the accuracy on the development set as the stopping criterion.

5 CRFs with Virtual Evidence

A canonical way of using virtual evidence (VE) in Bayesian networks is to have a directed edge from a hidden variable h to a VE variable v. The variable v will always be observed with a particular value, *e.g.*, $v = 1$, but the actual value itself does not matter. The prior knowledge about h is

[2] *Start* and *End* are with regard to a document, which are different from start and end of a sentence.

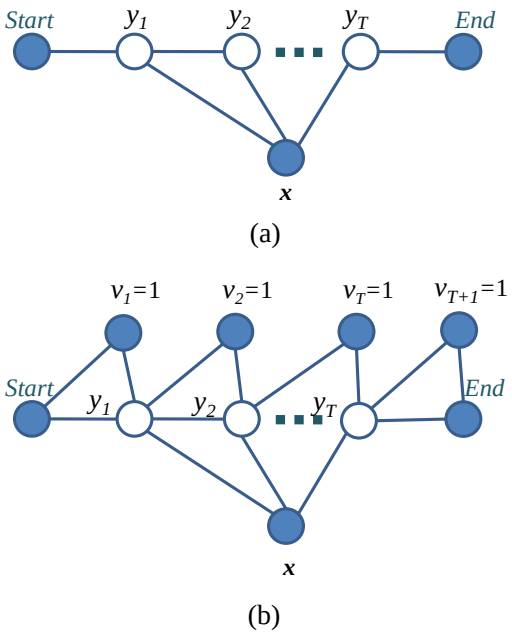


Figure 1: Graphical model representations of (a) a CRF model and (b) a CRF model integrated with virtual evidence. Solid and empty nodes denote observed and hidden variables respectively.

expressed via the conditional probability $p(v = 1|h)$. For example, by setting $p(v = 1|h = a) > p(v = 1|h = b)$, we know that $h = a$ is more likely a event than $h = b$. This conditional distribution is not learned from data, Instead, it is predefined in such a way that reflects a prior belief over the value of h.

VE can be encoded in an undirected graphical model in a similar fashion. For our task, we modify the structure of a linear-chain CRF model as depicted in Figure 1(b) — we create a sequence of VE variables, denoted by $v_1, v_2, \ldots, v_{T+1}$, in parallel to the state variables. Each v_t is assigned a constant 1 (one), and is connected with y_{t-1} and y_t, forming a new set of maximum cliques (y_{t-1}, y_t, v_t), $t = 1, \ldots, T + 1$. We create cliques of size 3 because it is the minimum size required to represent the prior knowledge used in our task, as will be discussed shortly. However, it is possible to have a different graph structure to incorporate other types of prior knowledge, *e.g.*, using large cliques to represent constraints that involve more variables.

Next, in analogy to Equation (5), we define the corresponding potential functions as follows,

$$\phi^{(t)}(y_{t-1}, y_t, v_t) = \exp\Big(\omega \cdot s(y_{t-1}, y_t, v_t, t)\Big) \tag{7}$$

s is a vector of VE feature functions and ω is the corresponding weight vector with pre-defined values. Given the new graphical model in Figure 1(b). It is natural to model the conditional probability of the state sequence given *both* the standard evidence and the VE as follows,

$$p_\lambda(\mathbf{y}|\mathbf{x}, \mathbf{v}) = \frac{1}{Z_\lambda(\mathbf{x}, \mathbf{v})} \prod_t \psi_\lambda^{(t)}(\mathbf{x}, y_{t-1}, y_t)\phi^{(t)}(y_{t-1}, y_t, v_t) \tag{8}$$

Analogous to using $p(v = 1|h)$ in Bayesian networks, we can utilize $\phi^{(t)}(y_{t-1}, y_t, \mathbf{v} = \mathbf{1})$ to express preferences over state hypotheses in a CRF model. In general, the function form of $\phi^{(t)}$ may or may not depend on the input $\mathbf{x}$. Even when $\phi^{(t)}$ does depend on $\mathbf{x}$, the relation is completely determined by external knowledge/systems (as opposed to by data). Thus we do not explicitly connect $\mathbf{v}$ with $\mathbf{x}$ in the graph.

5.1 Incorporating prior knowledge

Now we show how to represent the prior knowledge introduced in Section 3 using the VE feature functions. Unless otherwise stated, we assume $v_t = 1$ for all $t = 1, \ldots, T$ and simply use v_t instead of $v_t = 1$ in all equations. First, we define a VE function s_1 that represents K1: *label consistency with prototypes*. We let P_l denote a prototype list associated with the label l. If x_t belongs to P_l, we should prefer $y_t = l$ as opposed to other values. To this end, for cases where $x_t \in P_l$, we set s_1 as

$$s_1(y_t, v_t, t) = \begin{cases} 1 & \text{if } y_t = l \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

On the other hand, if x_t is not a prototype, we will always have $s_1(y_t, v_t, t) = 0$ for all hypotheses of y_t. The impact of this prior knowledge is controlled by the weight of s_1, denote by ω_1. At one extreme where $\omega_1 = 0$, the prior knowledge is completely ignored in training. At the other extreme where $\omega_1 \to +\infty$, we constrain the values of state variables to agree with the prior knowledge. Note that although s_1 is implicitly related to $\mathbf{x}$, we do not write s_1 as a function of $\mathbf{x}$ for consistency with the general definition of VE.

To represent K2: *label consistency within a sentence*, we define a second VE feature function s_2 with weight ω_2. Assume that we have an external system that detects sentence boundaries. If it is determined that x_t is *not* the start of a sentence, we set s_2 as

$$s_2(y_{t-1}, y_t, v_t, t) = \begin{cases} 1 & \text{if } y_{t-1} = y_t \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

It is easy to see that this would penalize state transitions within a sentence. On the other hand, if x_t is a sentence start, we set $s_2(y_{t-1}, y_t, v_t, t) = 0$ for all possible (y_{t-1}, y_t) pairs. In this work, we use a simple heuristics to detect sentence boundaries: we determine that x_t is the start of a sentence if its previous token x_{t-1} is a period (.), a semi-colon (;) or an acclamation mark (!), *and* if x_t is not a punctuation.

5.2 Semi-supervised learning

When a large amount of unlabeled data is available, it is often helpful to leverage such data to improve learning. However, we cannot directly optimize $p(\mathbf{y}|\mathbf{x}, \mathbf{v})$ since the correct state sequences of the unlabeled data are hidden. One heuristic approach is to adapt the *self-training* algorithm (Yarowsky, 1995) to our model. More specifically, for each input in the unlabeled dataset $\{\mathbf{x}^{(i)}\}_{i=m+1}^{n}$, we decode the best state sequence,

$$\hat{\mathbf{y}}^{(i)} = \underset{\mathbf{y}^{(i)}}{\text{argmax}}\ p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \mathbf{v}^{(i)}) \quad (11)$$

Then we use $\{(\mathbf{x}^{(i)}, \hat{\mathbf{y}}^{(i)})\}_{i=m+1}^{n}$ in addition to the labeled data to train a supervised CRF model. This approach, however, does not have a theoretical guarantee on optimality unless certain nontrivial conditions are satisfied (Abney, 2004).

On the other hand, it is well known that unlabeled data can be naturally incorporated using a generative approach that models a joint probability (Nigam et al., 2000). This is achieved by maximizing a marginal distribution of the joint probability over hidden variables. Inspired by the generative approach, we propose to explicitly model $p(\mathbf{y}, \mathbf{v}|\mathbf{x})$. In contrast to Equation (8), here we jointly model $\mathbf{y}$ and $\mathbf{v}$ but the probability is still conditioned on $\mathbf{x}$. This "joint" distribution should be chosen such that it results in the same conditional distribution $p(\mathbf{y}|\mathbf{x}, \mathbf{v})$ as defined in Equation (8). To this end, we define $p_\lambda(\mathbf{y}, \mathbf{v}|\mathbf{x})$ as

$$p_\lambda(\mathbf{y}, \mathbf{v}|\mathbf{x}) = \frac{1}{Z'_\lambda(\mathbf{x})} \prod_t \psi_\lambda^{(t)}(\mathbf{x}, y_{t-1}, y_t)\phi^{(t)}(y_{t-1}, y_t, v_t) \quad (12)$$

Here $Z'_\lambda(\mathbf{x})$ is a normalization function obtained by summing the numerator over both $\mathbf{y}$ and $\mathbf{v}$. By applying the Bayes rule, it is easy to see that $p(\mathbf{y}|\mathbf{x}, \mathbf{v})$ is exactly equal to Equation (8).

Given unlabeled data $\{\mathbf{x}^{(i)}\}_{i=m+1}^{n}$, we aim to optimize the following objective,[3]

$$\mathcal{L}_2 = \sum_{i=1}^{m+n} \log p_\lambda(\mathbf{v}^{(i)}|\mathbf{x}^{(i)}) - \frac{1}{2\sigma^2}\|\lambda\|^2 \quad (13)$$

This is essentially the marginal distribution of $p(\mathbf{y}, \mathbf{v}|\mathbf{x})$ over hidden variables $\mathbf{y}$. Here we ignore the labels of the dataset $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m}$, but we do use the label information in initializing the model which will described in Section 6. To optimize such an objective, we apply the EM algorithm in the same fashion as is used in a generative approach. In other words, we iteratively optimize $Q(\lambda) = \sum_{\mathbf{y}} p_{\lambda^g}(\mathbf{y}|\mathbf{x}, \mathbf{v}) \log p_\lambda(\mathbf{y}, \mathbf{v}|\mathbf{x})$ where λ^g denotes the model estimated from the previous iteration. The gradient of the Q function is straightforward to compute with the result given by

$$\frac{\partial Q(\lambda)}{\partial \lambda_k} = \sum_t \sum_{y_{t-1}, y_t} f_k(y_{t-1}, y_t, \mathbf{x}, t) \cdot \Big(p_\lambda(y_{t-1}, y_t|\mathbf{x}, \mathbf{v}) - p_\lambda(y_{t-1}, y_t|\mathbf{x}) \Big) \quad (14)$$

We keep two sets of accumulators in running the Forward-Backward algorithm, one for computing $p_\lambda(y_{t-1}, y_t|\mathbf{x}, \mathbf{v})$ and the other for computing $p_\lambda(y_{t-1}, y_t|\mathbf{x})$. Loosely speaking, the model will converge to a local optimum if the difference between these two posterior probabilities becomes trivial.

5.3 Collocation based virtual evidence

Prior knowledge represented by prototypes is typically sparse. This sparse information, however, can be propagated across all data based on distributional similarity (Haghighi and Klein, 2006). Following the same idea, we extend the prototype lists as follows. (1) We merge all prototypes in P_l into a single word type w_l. (2) For each word

[3] In Equation (13), the fact that $\mathbf{v}$ is assigned a constant 1 does not mean $p(\mathbf{v} = \mathbf{1}|\mathbf{x}) = 1$ (Bilmes, 2004)

Label	Prototype lists of HK06	Collocation lists (top examples)
ADDRESS	address carlmont	[4-digit] street [3-digit] streets
AVAILABLE	immediately begin cheaper	available
CONTACT	[phone] call [time]	[email] appointment email see today ...
FEATURES	kitchen laundry parking	room new covered building garage ...
NEIGHBORHOOD	close near shopping	transportation center located restaurants ...
PHOTOS	pictures image link	[url] click view photos
RENT	$ month [amount]	lease deposit security year agreement ...
RESTRICTIONS	pets smoking dog	ok sorry please allowed negotiable ...
ROOMMATES	roommate respectful drama	
SIZE	[1-digit] br sq	[4-digit] [3-digit] ft bath ba ...
UTILITIES	utilities pays electricity	water included owner garbage paid

Table 1: Field labels (except *other*) for the CLASSIFIEDS task, their respective prototype lists specified by prior knowledge, and collocation lists mined from unlabeled data.

in the corpus, we collect a context vector of the counts of all words (excluding stop words) that occur within a window of size k in either direction, where the window is applied only within sentence boundaries. (3) Latent semantic analysis (Deerwester et al., 1990) is performed on the constructed context vectors. (4) In the resulting latent semantic space, all words (except stop words) that have a high enough dot product with w_l will be grouped to form a new set, denoted as C_l, which is a superset of P_l. In this regard, C_l can be viewed as lists of noisy "prototypes". As observed in HK06, another consequence of this method is that many neighboring tokens will share the same prototypes.

Differently from previous works, we use C_l directly as virtual evidence. We could apply s_1 in Equation (9) when $x_t \in C_l$ (as opposed to when $x_t \in P_l$). This, however, would contaminate our model since C_l are often noisy. For example, "water" is found to be distributionally similar to the prototypes of *utilities*. Although in most cases "water" indeed means *utilities*, it can mean *features* in the context of "water front view". To maximally reduce ambiguity, we propose to apply s_1 in Equation (9) if *both* of the following conditions hold,

(1) $x_t \in C_l$

(2) There exists τ s.t. $|\tau - t| < k$, and $x_\tau \in C_l$

In other words, we will impose a non-uniform prior on y_t if $x_t \in C_l$ "collocates", within k tokens, with another word that belongs to C_l. Based on K2, it is reasonable to believe that neighboring tokens tend to share the same label. Therefore, knowing that two tokens close to each other both belong to C_l would strengthen our belief that either word is likely to have label l. We thus refer to this type of virtual evidence as *collocation-based VE*, and refer to C_l as *collocation lists*.

6 Evaluation

We use the CLASSIFIEDS data provided by Grenager et al. (2005) and compare with results reported by CRR07 (Chang et al., 2007) and MM08 (Mann and McCallum, 2008) for both supervised and semi-supervised learning. Following all previous works conducted on this task, we tokenized both words and punctuations, and created a number of regular expression tokens for phone numbers, email addresses, URLs, dates, money amounts and so on. However, we did not tokenize newline breaks, as CRR07 did, which might be useful in determining sentence boundaries. Based on such tokenization, we extract n-grams, $n = 1, 2, 3$, from the corpus as features for CRFs.

As described in Section 3, we integrate the prior knowledge K1 and K2 in our CRF model. The prototypes that represent K1 are given in Table 1. CRR07 used the same two kinds of prior knowledge in the form of constraints, and they implemented another constraint on the minimum number of words in a field chunk. MM08 used almost the same set of prototypes as labeled features, but they exploited two sets of 33 additional features for some experiments. In this regard, the comparison between CRR07, MM08 and the method presented here cannot be exact. However, we show that while our prior knowledge is no more than that used in previous works, our approach is able

Supervised model	# labeled examples 10	25	100
CRR07: HMM	61.6	70.0	76.3
+ Constr in decoding	66.1	73.7	80.4
MM08: CRF	64.6	72.9	79.4
CRF	62.3	71.4	79.1
+ VE in decoding	**68.9**	**74.6**	**81.1**
CRF + VE (auto weights)	48.0	54.8	59.8
+ VE in decoding	66.0	72.5	80.9

Table 2: Token-level accuracy of supervised learning methods; "+ VE" refers to the cases where both kinds of prior knowledge, K1 and K2, are incorporated as VE in the CRF model.

Semi-supervised models	# labeled examples 10	25	100
CRR07: HMM + Constr	70.9	74.8	78.6
+ Constr in decoding	74.7	78.5	81.7
MM08: CRF + GE	72.6	76.3	80.1
CRF + VE (Self-train)	69.0	74.2	81.4
+ VE in decoding	69.1	75.2	81.2
CRF + Col-VE (Self-train)	73.1	76.4	81.8
+ Col-VE in decoding	75.7	77.6	82.9
CRF + Col-VE (EM)	78.3	79.1	82.7
+ Col-VE in decoding	**78.8**	**79.5**	**82.9**

Table 3: Token-level accuracy of semi-supervised learning methods. "+ Col-VE" refers to cases where collocation-based VE is integrated in the CRF model in addition to the VE representing K1 and K2.

to achieve the state-of-art performance.

6.1 Decoding settings

Depending on whether VE is used at test time, we explore two decoding settings in all experiments:

1. Find $\mathbf{y}$ that maximizes $p_\lambda(\mathbf{y}|\mathbf{x})$ as in standard CRF decoding, ignoring virtual evidence.
2. Find $\mathbf{y}$ that maximizes $p(\mathbf{y}|\mathbf{x}, \mathbf{v})$. We use "+ VE in decoding" to represent this setting.

These two scenarios are analogous to those in CRR07 which conducted HMM decoding without/with constraints applied. We use "+ constr. in decoding" to represent the latter scenario of their work. MM08, on the other hand, found no accuracy improvement when adding constraints at test time.

Note that in our second decoding setting, the weights for the VE feature functions, *i.e.*, ω_1 and ω_2, are tuned on the development set. This is done by a greedy search that first finds the best ω_1, and then finds the best ω_2 while fixing the value of ω_1, both with a step size 0.5.

6.2 Supervised learning results

First, we experimented with a standard CRF model with VE applied neither in training nor in decoding. As shown in Table 2, our CRF implementation performed slightly worse than the implementation by MM08, probably due to slight difference in tokenization. Secondly, we used the same CRF model but additionally applied VE in decoding, corresponding to the second setting in Section 6.1. This method gave a significant boost to the tagging performance, yielding the best supervised learning results (shown as bolded in the table). This proves that the prior knowledge is indeed complementary to the information offered by the training data.

Similar to the second decoding setting that incorporates VE, we can have a counterpart setting at training time. In other words, we can optimize $p_\lambda(\mathbf{y}|\mathbf{x}, \mathbf{v})$ instead of $p_\lambda(\mathbf{y}|\mathbf{x})$ during learning. In deciding $\omega = (\omega_1, \omega_2)$, it is possible to learn ω from data in the same way as how we learn λ. This, however, might undermine the role of other useful features since we do not always have sufficient training data to reliably estimate the weight of prior knowledge. As shown in Table 2, we experimented with learning ω automatically (shown as "auto weights"). While applying VE with such weights in both training and decoding worked reasonably well, applying VE only in training but not in decoding yielded very poor performance (probably due to excessively large estimates of ω_1 and ω_2). Additionally, we repeated the above experiment with manually specified weights, but did not find further accuracy improvement over the best supervised learning results.

6.3 Semi-supervised learning results

One natural way of leveraging the unlabeled data (more than 8K examples) is to perform semi-supervised learning in a self-training fashion. To this end, we used our best supervised model in Table 2 to decode the unlabeled examples as well as the test-set examples (by treating them as unlabeled). Note that by doing this our comparison with CRR07 and MM08 cannot be exact as they

sampled the unlabeled examples, with different rates, for semi-supervised learning, while we used as much data as possible. We applied the same ω that was used for the supervised model, and then combined the newly labeled examples, in addition to the manually labeled ones, as training data to learn a supervised CRF model. On this particular dataset, we did not find it helpful by selecting automatically labeled data based on a confidence threshold. We simply used all data available in self-training. This paradigm is referred to as "CRF + VE (self-train)" in Table 3. When no VE is applied at test time, this semi-supervised CRF model significantly outperformed the best model in Table 2. When applying VE at test time, however, the improvement over its supervised counterpart became trivial.

Next, following Section 5.3, we collected context vectors on the unlabeled data using a window size $k = 3$, and extracted the top 50 singular vectors therefrom.[4] We created collocation lists that contain words close to the merged prototype words in the latent semantic space. Some examples are given in the last column of Table 1. We then augmented the prototype-based VE based on the following rules: If x_t belongs to any prototype list P_l, we directly apply s_1 in Equation (9); otherwise, we apply s_1 if x_t *and* at least one neighbor (within 3 tokens from x_t) belong to the same collocation list C_l. In our experiments, we let "Col-VE" represent such collocation-based VE. We conducted self-training using a CRF model integrated with Col-VE, where ω was tuned *a priori* by testing the same model on the development set. As shown in the table, "CRF + Col-VE (self-train)" gave significant accuracy improvement over "CRF + VE", while adding Col-VE at test time further boosted the performance. The accuracies were already on par with the best results previously reported on this task.

Finally, we implemented the EM algorithm proposed in Section 5.2 that iteratively optimizes $p(\mathbf{v}|\mathbf{x})$ on all data. The model was initialized by the one obtained from "CRF + Col-VE (self-train)". After the model was initialized, we performed the EM algorithm until the model reached a maximum accuracy on the development set. Note that in some cases, we observed a development-set accuracy degradation after the first iteration of the EM, but the accuracy quickly recovered from the second iteration and kept increasing until a maximum accuracy was reached.[5] As shown in the last two rows in Table 3, this method is clearly advantageous over self-training, leading to the best tagging accuracies in both decoding settings. Our model achieved $2.6\% - 5.7\%$ absolute accuracy increases in the three training settings compared with MM08 which had the best results without using any constraints in decoding. When applying VE at test time, our model was $1.2\% - 4.1\%$ better than CRR07 which had the best overall results. Additionally, when compared with supervised learning results, our best semi-supervised model trained on only 10 labeled examples performed almost as well as a standard supervised CRF model trained on 100 labeled examples.

[4]The same configuration is used in HK06.

7 Conclusions

We have presented the use of virtual evidence as a principled way of incorporating prior knowledge into conditional random fields. A key contribution of our work is the introduction of a novel semi-supervised learning objective for training a CRF model integrated with VE. We also found it useful to create so-called collocation-based VE, assuming that tokens close to each other tend to have consistent labels. Our evaluation on the CLASSIFIEDS data showed that the learning objective presented here, combined with the use of collocation-based VE, yielded remarkably good accuracy performance. In the future, we would like to see the application of our approach to other tasks such as (Li et al., 2009).

References

Steven Abney. 2004. Understanding the Yarowsky algorithm. *Association for Computational Linguistics*, 30(3):365–395.

Jeff Bilmes. 2004. On soft evidence in Bayesian networks. Technical Report UWEETR-2004-0016, University of Washington, Dept. of Electrical Engineering.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proceedings of ACL*.

[5]The initial degradation is probably due to the fact that self-training can result in an initial model with decent accuracy but low $p(\mathbf{v}|\mathbf{x})$; thus the EM algorithm that maximizes $p(\mathbf{v}|\mathbf{x})$ may temporarily decrease the accuracy.

Scott Deerwester, Susan Dumais, Thomas Landauer, George Furnas, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

Yves Grandvalet and Yoshua Bengio. 2004. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*.

Trond Grenager, Dan Klein, and Christopher D. Manning. 2005. Unsupervised learning for field segmentation models for information extraction. In *Proceedings of Association of Computational Linguistics*.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of HLT-NAACL*.

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of ACL*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289.

Xiao Li, Ye-Yi Wang, and Alex Acero. 2009. Extracting structured information from user queries with semi-supervised conditional random fields. In *Proceedings of SIGIR*.

Gideon Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of ACL*.

Kamal Nigam, Andrew Kachites Mccallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2nd printing edition edition.

A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell. 2007. Hidden conditional random fields. *IEEE Transaction on Pattern Analysis and Machine Intellegence*, 29(10):1848–1852.

Sheila Reynolds and Jeff Bilmes. 2005. Part-of-speech tagging using virtual evidence and negative training. In *Proceedings of HLT/EMNLP*.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using gigaword scale unlabeled data. In *Proceedings of ACL/HLT*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL*, pages 189–196.

Refining Grammars for Parsing with Hierarchical Semantic Knowledge

Xiaojun Lin, Yang Fan, Meng Zhang, Xihong Wu*, Huisheng Chi
Speech and Hearing Research Center
Key Laboratory of Machine Perception (Ministry of Education)
School of Electronics Engineering and Computer Science
Peking University, Beijing, 100871, China
{linxj, fanyang, zhangm, wxh}@cis.pku.edu.cn, chi@pku.edu.cn

Abstract

This paper proposes a novel method to refine the grammars in parsing by utilizing semantic knowledge from HowNet. Based on the hierarchical state-split approach, which can refine grammars automatically in a data-driven manner, this study introduces semantic knowledge into the splitting process at two steps. Firstly, each part-of-speech node will be annotated with a semantic tag of its terminal word. These new tags generated in this step are semantic-related, which can provide a good start for splitting. Secondly, a knowledge-based criterion is used to supervise the hierarchical splitting of these semantic-related tags, which can alleviate overfitting. The experiments are carried out on both Chinese and English Penn Treebank show that the refined grammars with semantic knowledge can improve parsing performance significantly. Especially with respect to Chinese, our parser achieves an F_1 score of 87.5%, which is the best published result we are aware of.

1 Introduction

At present, most high-performance parsers are based on probabilistic context-free grammars (PCFGs) in one way or another (Collins, 1999; Charniak and Johnson, 2005; Petrov and Klein, 2007). However, restricted by the strong context-free assumptions, the original PCFG model which simply takes the grammars and probabilities off a treebank, does not perform well. Therefore, a variety of techniques have been developed to enrich and generalize the original grammar, ranging from lexicalization to symbol annotation.

*Corresponding author: Xihong Wu.

Lexicalized PCFGs use the structural features on the lexical head of phrasal node in a tree, and get significant improvements for parsing (Collins, 1997; Charniak, 1997; Collins, 1999; Charniak, 2000). However, they suffer from the problem of fundamental sparseness of the lexical dependency information. (Klein and Manning, 2003).

In order to deal with this limitation, a variety of unlexicalized parsing techniques have been proposed. Johnson (1998) annotates each node by its parent category in a tree, and gets significant improvements compared with the original PCFGs on the Penn Treebank. Then, some manual and automatic symbol splitting methods are presented, which get comparable performance with lexicalized parsers (Klein and Manning, 2003; Matsuzaki et al., 2005). Recently, Petrov et al. (2006) introduces an automatic hierarchical state-split approach to refine the grammars, which can alternately split and merge the basic nonterminals by the Expectation-Maximization (EM) algorithm. In this method, the nonterminals are split to different degrees, as appropriate to the actual complexity in the data. The grammars refined in this way are proved to be much more accurate and compact than previous work on automatic annotation. This data-driven method still suffers from the overfitting problem, which may be improved by integrating other external information.

In this paper, we propose a novel method that combines the strengths of both data-driven and knowledge-driven strategies to refine grammars. Based on the work proposed by Petrov et al. (2006), we use the semantic knowledge from HowNet (Dong and Dong, 2000) to supervise the hierarchical state-split process at the part-of-speech(POS) level. At first, we define the most general hypernym in HowNet as the semantic class of a word, and then use this semantic class to initialize the tag of each POS node. In this way, a new set of semantic-related tags is generated, and

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1298–1307,
Singapore, 6-7 August 2009.

a good starting annotation is provided to reduce the search space for the EM algorithm in the splitting process. Then, in order to mitigate the overfitting risk, the hierarchical hypernym-hyponym relation between hypernyms in HowNet is utilized to supervise the splitting of these new semantic-related tags. By introducing a knowledge-based criterion, these new tags are decided whether or not to split into subcategories from a semantic perspective. To investigate the effectiveness of the presented approach, several experiments are conduced on both Chinese and English. They reveal that the semantic knowledge is potentially useful to parsing.

The remainder of this paper is organized as follows. Section 2 reviews some closely related works, including the lexical semantic related parsing and the hierarchical state-split unlexicalized parsing. In section 3, the presented method for grammar refining is described in detail, and several experiments are carried out for evaluation in Section 4. Conclusions are drawn in Section 5.

2 Background

This paper tries to refine the grammars through an improved hierarchical state-split process integrated with semantic knowledge. The related works are reviewed as follows.

2.1 Lexical Semantic Related Parsing

Semantic knowledge is useful to resolving syntactic ambiguities, and a variety of researches focus on how to utilize it. Especially in recent years, a conviction arose that semantic knowledge could be incorporated into the lexicalized parsing.

Based on the lexicalized grammars, Bikel (2000) attempts at combining parsing and word sense disambiguation in a unified model, using a subset of SemCor (Miller et al., 1994). Bikel (2000) evaluates this model in a parsing context with sense information from WordNet, but does not get improvements on parsing performance.

Xiong et al. (2005) combines word sense from CiLin and HowNet (two Chinese semantic resources) in a generative parsing model, which generalizes standard bilexical dependencies to word-class dependencies, and indeed help to tackle the sparseness problem in lexicalized parsing. The experiments show that the parse model combined with word sense and the most special hypernyms achieves a significant improvement on Penn Chinese Treebank. This work only considers the most special hypernym of a word, rather than other hypernyms at different levels of the hypernym-hyponym hierarchy.

Then, Fujita et al. (2007) uses the Hinoki treebank as training data to train a discriminative parse selection model combining syntactic features and word sense information. Instead of utilizing the most special hypernym, the word sense information in this model is embodied with more general concepts. Based on the hand-craft sense information, this model is proved to be effective for parse selection.

Recently, Agirre et al. (2008) train two lexicalized models (Charniak, 2000; Bikel, 2004) on preprocessed inputs, where content words are substituted with semantic classes from WordNet. By integrating the word semantic classes into the process of parser training directly, these two models obtain significant improvements in both parsing and prepositional phrase attachment tasks. Zhang (2008) does preliminary work on integrating POS with semantic class of words directly, which can not only alleviate the confusion in parsing, but also infer syntax and semantic information at the same time.

2.2 The Hierarchical State-split Parsing

In order to alleviate the context-free assumptions, Petrov et al. (2006) proposes a hierarchical state-split approach to refine and generalize the original grammars, and achieves state-of-the-art performance. Starting with the basic nonterminals, this method repeats the split-merge (SM) cycle to increase the complexity of grammars. That is, it splits every symbol into two, and then re-merges some new subcategories based on the likelihood computation.

Splitting

In each splitting stage, the previous syntactic symbol is split into two subcategories, and the EM algorithm is adopted to learn probability of the rules for these latent annotations to maximize the likelihood of trees in the training data. Finally, each symbol generates a series of new subcategories in a hierarchical fashion. With this method, the splitting strategy introduces more context information, and the refined grammars cover more linguistic information which helps resolve the syntactic ambiguities.

However, it is worth noting that the EM algorithm does not guarantee a global optimal solution, and often gets stuck in a suboptimal configuration. Therefore, a good starting annotation is expected to help alleviate this problem, as well as reduce the search space for EM.

Merging

It is obvious that using more derived subcategories can increase accuracy, but the refined grammars fit tighter to the training data, and may lead to overfitting to some extent. In addition, different symbols should have their specific numbers of subcategories. For example, the comma POS tag should have only one subcategory, as it always produces the terminal comma. On the contrary, the noun POS tag and the verb POS tag are expected to have much more subcategories to express their context dependencies. Therefore, it is not reasonable to split them in the same way.

The symbol merging stage is introduced to alleviate this defect. This approach splits symbols only where needed, and it is implemented by splitting each symbol first and then measure the loss in likelihood incurred when removing this subcategory. If the loss is small, it means that this subcategory does not take enough information and should be removed. In general it is hard to decide the threshold of the likelihood loss, and this merging stage is often executed by removing a certain proportion of subcategories, as well as giving priority to the most informative subcategories.

By splitting and merging alternately, this method can refine the grammars step by step to mitigate the overfitting risk to some extent. However, this data-driven method can not solve this problem completely, and we need to find other external information to improve it.

Analysis

The hierarchical state-split approach is used to split all the symbols in the same way. Table 1 cites the subcategories for several POS tags, along with their two most frequent words. Results show that the words in the same subcategory of POS tags are semantic consistent in some cases. Therefore, it is expected to optimize the splitting and merging process at the POS level with semantic knowledge.

NR

NR-0	大甲溪(Daja river)	尼泊尔(Nepal)
NR-1	新力(Sony)	伯乐网(Bole Co.)
NR-2	华诚(C. Hua)	文天祥(T. Wen)
NR-3	乐绍延(S. Yue)	商(Shang)

LC

LC-0	当中(middle)	右侧(right)
LC-1	以前(before)	以来(since)
LC-2	开始(start)	止(end)
LC-3	为止(till)	末(end)

P

P-0	每当(whenever)	至于(as for)
P-1	犹如(like)	仿佛(as)
P-2	朝着(look to)	照着(according to)
P-3	傍(be close to)	比照(contrast)

Table 1: The two most frequent words in the subcategories of several POS tag.

3 Integration with Semantic Knowledge

In this paper, the semantic knowledge is used to refine grammars by improving the automatic hierarchical state-split approach. At first, in order to provide good starting annotations to reduce the search space for the EM algorithm, we try to annotate the tag of each POS node with the most general hypernym of its terminal word. In this way, we generate a new set of semantic-related tags. And then, instead of splitting and merging all symbols together automatically, we propose a knowledge-based criterion with hierarchical semantic knowledge to supervise the splitting of these new semantic-related tags.

3.1 HowNet

The semantic knowledge resource we use is HowNet, which is a common sense knowledge base unveiling concepts and inter-conceptual relations in Chinese and English.

As a knowledge base of graph structure, HowNet is devoted to demonstrating the properties of concepts through sememes and relations between sememes. Broadly speaking, a sememe refers to the smallest basic semantic unit that cannot be reduced further, which can be represented in English and their Chinese equivalents, such as the sememe *institution*|*机构*. The relations explicated in HowNet include hypernym-hyponym relations, location-event relations, time-event relations and so on. In this work, we mainly focus on

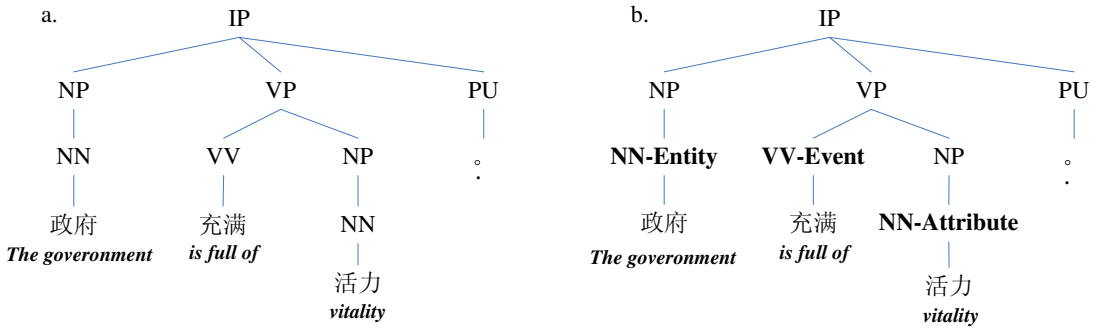


Figure 1: The two syntax trees of the sentence "*The government is full of vitality*". a. is the original syntax tree, b. is the syntax tree in which each tag of the POS node is annotated with the most general hypernym of its terminal word.

the hypernym-hyponym relations. Take the word 政府(government) as an example, its hypernyms with the hierarchical hypernym-hyponym relations are listed below from speciality to generality, which we call hierarchical semantic information in this paper.

institution|机构→group|群体→thing|万物→entity|实体

It is clear that this word 政府(government) has hypernyms from the most special hypernym *institution|机构* to the most general hypernym *entity|实体* in a hierarchical way.

In HowNet(Update 2008), there are 173535 concepts, with 2085 sememes. The sememes are categorized into entity, event, attribute, attribute value, etc., each corresponding to a sememe hierarchy tree.

3.2 Annotating the Training Data

One of the original motivations for the grammar refinement is that the original symbols, especially the POS tags, are usually too general to distinguish the context dependencies. Take the sentence in Figure 1 for example, the word 政府(government) should have different context dependencies compared with the word 活力(vitality), although both of them have the same POS tag "NN". In fact, the two words are defined in HowNet with different hypernyms. The word 政府(government) is defined as a kind of objective things, while the word 活力(vitality) is defined as a property that is often used to describe things. It is obvious that the different senses can represent their different syntax structures, and we expect to refine the POS tags with semantic knowledge.

In the automatic hierarchical state-split approach introduced above, the EM algorithm is used to search for the maximum of the likelihood during the splitting process, which can generate subcategories for POS tags to express the context dependencies. However, this method often gets stuck in a suboptimal configuration, which varies depending on the start point. Therefore, a good start of the annotations is very important. As it is displayed in Figure 1, we annotate the tag of each POS node with the hypernym of its terminal word as the starting annotation. There are two problems that we have to consider in this process: a) how to choose the appropriate semantic granularity, and b) how to deal with the polysemous words.

As mentioned above, the semantic information of each word can be represented as hierarchical hypernym-hyponym relations among its hypernyms. In general, it is hard to decide the appropriate level of granularity to represent the word. The semantic class is only used as the starting annotations of POS tags to reduce the search space for EM in our method. It is followed by the hierarchical state-split process to further refine the starting annotations based on the structural information. If more special kinds of semantic classes are chosen, it will make the structural information weaker. As annotations with the special hypernym always defeat some of the advantage of automatically latent annotations learning, we annotate the training data with the most general hypernym. For example, as shown in Figure 1, the POS tag "NN" of 政府(government) is annotated as "NN-Entity", and "NN" of 活力(energy) is annotated as "NN-Attribute".

Another problem is how to deal with the polysemous words in HowNet. In fact, when we choose the most general hypernym as the word's semantic

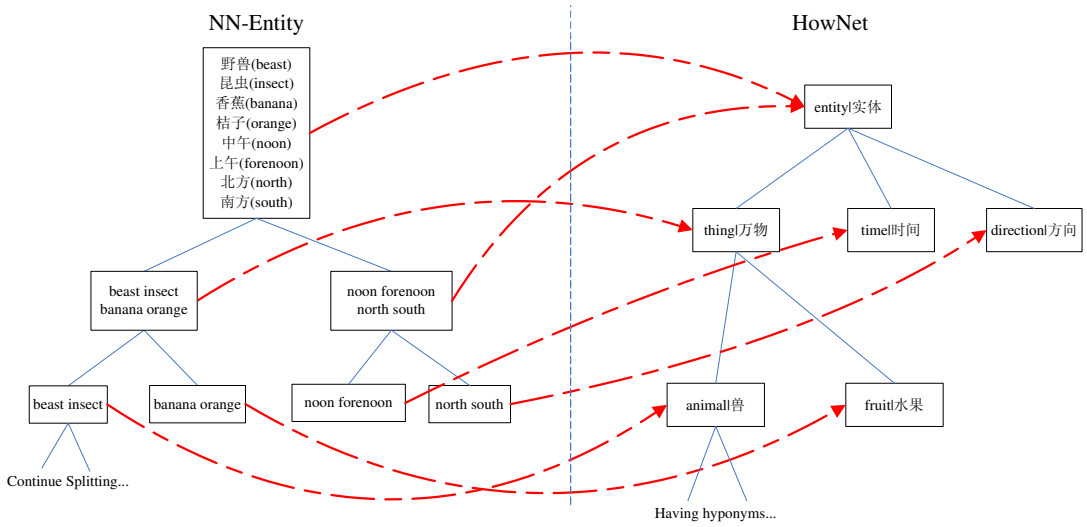


Figure 2: A schematic figure for the hierarchical state-split process of the semantic-related tag "NN-Entity". Each subcategory of this tag has its own word set, and corresponds to one hypernym at the appropriate level in HowNet.

representation, this problem has been alleviated to a large extent. In this paper we adopt the first sense option as our word sense disambiguation (WSD) strategy to determine the sense of each token instance of a target word. That is to say, all token instances of a given word are tagged with the sense that occurs most frequently in HowNet. In addition, we keep the tag of the POS node whose terminal word is not defined in HowNet unchanged.

3.3 Supervising the Hierarchical State-split Process

With the method proposed above, we can produce a good starting annotation with semantic knowledge, which is of great use to constraining the automatic splitting process. Our parser is trained on the good starting annotations with the automatic hierarchical state-split process, and gets improvements compared with the original training data. However, during this process, only the most general hypernyms are used as the semantic representation of words, and the hierarchical semantic knowledge is not explored. In addition, the automatic process tries to refine all symbols together through a data-driven manner, which suffers the overfitting risk.

After annotating the training data with hypernyms, a new set of semantic-related tags such as "NN-Entity" is produced. We treat the refining process of these semantic-related tags as the specializing process of hypernym with hierarchical semantic knowledge. Each subcategory of these tags corresponds to a appropriate special level of hypernym in the HowNet. For example, every subcategory of "NN-Entity" could corresponds to a appropriate hyponym of *entity|实体*.

We integrate the hierarchical semantic knowledge into the original hierarchical state-split process to refine these semantic-related tags. First of all, it is necessary to establish the mapping from each subcategory of these semantic-related tags to the hypernym at the appropriate level in HowNet. Then, instead of likelihood judgment, a knowledge-based criterion is proposed, to decide whether or not to remove the new subcategories of these tags. That is to say, once the parent tag of this new subcategory is mapped onto the most special hypernym without any hyponym, it should be removed immediately.

The schematic Figure 2 demonstrates this semantically supervised splitting process. The left part of this figure is the subcategories of the semantic-related tag "NN-Entity", which is split hierarchically. As expressed by the dashed line, each subcategory corresponds to one hypernym in the right part of this figure. If the hypernym node has no hyponym, the corresponding subcategory will stop splitting.

The mapping from each subcategory of these semantic-related tags to the hypernym at the appropriate level is implemented with the word set related to this subcategory. As it is shown in Fig-

DataSet	Chinese Xue et al. (2002)	English Marcus et al. (1993)
TrainSet	Art. 1-270,400-1151	Sections 2-21
DevSet	Articles 301-325	Section 22
TestSet	Articles 271-300	Section 23

Table 2: Experimental setup.

ure 2, the original tag "NN-Entity" treats all the words it products as its word set. Once the original category is split into two subcategories, its word set is also split, through forcedly dividing each word in the word set into one subcategory which is most frequent with this word. And then, each subcategory is mapped onto the most specific hypernym that contains its related word set entirely in HowNet. On this basis, a new knowledge-based criterion is introduced to enrich and generalize these semantic-related tags, with purpose of fitting to the hierarchical semantic structure rather than the training data.

4 Experiments

In this section, we designed several experiments to investigate the validity of refining grammars with semantic knowledge.

4.1 Experimental Setup

We did experiments on Chinese and English. In order to make a fair comparison with previous works, we split the standard corpora as shown in Table 2. Our parsers were evaluated by the EVALB parseval reference implementation[1]. The Berkeley parser[2] was used to train the models with the original automatic hierarchical state-split process. The semantic resource we used to improve parsing was HowNet, which has been introduced in Subsection 3.1. Statistical significance was checked using Dan Bikel's randomized parsing evaluation comparator with the default setting of 10,000 iterations[3].

4.2 Semantic Representation Experiments

First of all, we ran experiments with different semantic representation methods on Chinese. The polysemous words in the training set were annotated with the WSD strategy of first sense option, which was proved to be useful in Agirre et al. (2008).

As mentioned in Subsection 3.1, the semantic information of each word can be represented as a hierarchical relation among its hypernyms from specialty to generalization in HowNet. In order to choose the appropriate level of granularity to represent words, we annotated the training set with different levels of granularity as semantic representation. In our experiments, the automatic hierarchical state-split process is used to train models on these training sets with different level of semantic representation.

We tried two kinds of semantic representations, one is using the most general hypernym, and the other is using the most special hypernym. Results in Figure 3 proved the effectiveness of our method in Subsection 3.2. When we annotated the tag of each POS node with the most general hypernym of its terminal word, the parser performs much better than both the baseline and the one annotated with the most special hypernym. Moreover, the F_1 score starts dropping after 3 training iterations on the training set annotated with the most special hypernyms, while it is still improving with the most general one, indicating overfitting.

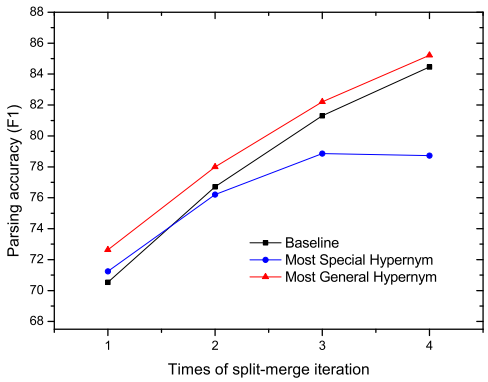


Figure 3: Performances on Chinese with different semantic representations: the training set without semantic representation, the training set annotated with the most special hypernyms, and the training set annotated with the most general hypernyms.

When the training set was annotated with the most general hypernyms, there were only 57 new semantic-related tags such as "NN-Entity", "NN-Attribute" and so on. However, when the training set was annotated with the most special hypernyms, 4313 new tags would be introduced. Ob-

[1] http://nlp.cs.nyu.edu/evalb/.
[2] http://code.google.com/p/berkeleyparser/.
[3] http://www.cis.upenn.edu/ dbikel/software.html.

viously, it introduces too many tags at once and is difficult to refine appropriate grammars in the subsequent step starting with this over-splitting training set.

4.3 Grammar Refinement Experiments

Several experiments were carried out on Chinese and English to verify the effectiveness of refining grammars with semantic knowledge. We took the most general hypernym as the semantic representation, and the polysemous words in the training set were annotated with the WSD strategy of first sense option.

In our experiments, three kinds of method were compared. The baseline was trained on the raw training set with the automatic hierarchical state-split approach. Then, we improved it with the semantic annotation, which annotated the raw training set with the most general hypernyms as semantic representations, while keeping the training approach used in the baseline unchanged. Further, our knowledge-based criterion was introduced to supervise the automatic hierarchical state-split process with semantic knowledge.

In this section, since most of the parsers (including the baseline parser and our advanced parsers) had the same behavior on development set that the accuracy continued increasing in the five beginning iterations and then dropped at the sixth iteration, we chose the results at the fifth iteration as our final test set parsing performance.

Performances on Chinese

Figure 4 shows that refining grammars with semantic knowledge can help improve parsing performance significantly on Chinese (sentences of length 40 or less). Benefitting from the good starting annotations, our parser achieved significant improvements compared with the baseline (86.8% vs. 86.1%, p<.08). It proved that the good starting annotations with semantic knowledge were effective in the splitting process. Further, we supervised the splitting of the new semantic-related tags from the semantic annotations, and achieved the best results at the fifth iteration. The best F_1 score reached 87.5%, with an error rate reduction of 10.1%, relative to the baseline (p<.004).

Table 3 compared our methods with the best previous works on Chinese. The result showed that refining grammars integrated with semantic knowledge could resolve syntactic ambiguities remarkably and achieved the state-of-the-art performance on Chinese.

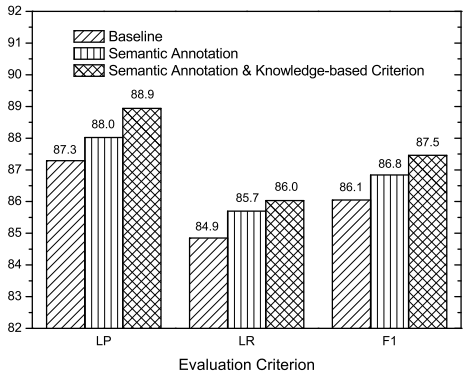


Figure 4: Performances at the fifth iteration on Chinese (sentences of length 40 or less) with three methods: the baseline, the parser trained on the semantic annotations with automatic method, and the parser trained on the semantic annotations with knowledge-based criterion.

Parser	≤ 40 words			all		
	LP	LR	F_1	LP	LR	F_1
Chiang and Bikel (2002)	81.1	78.8	79.9	78.0	75.2	76.6
Petrov and Klein (2007)	86.9	85.7	86.3	84.8	81.9	83.3
This Paper	**88.9**	**86.0**	**87.5**	**86.0**	**83.1**	**84.5**

Table 3: Our final parsing performance compared with the best previous works on Chinese.

Performances on English

In order to verify the effectiveness of our method on other languages, we carried out some experiments on English. HowNet is a common sense knowledge base in Chinese and English, therefore, it was still utilized as the knowledge source in these experiments.

The same three methods were compared on English (sentences of length 40 or less), and the results were showed in Table 4. Compared with the baseline (90.1%), the parsers trained with the semantic annotation, while using different splitting methods introduced in Section 3, achieved an F_1 score of 90.2% and 90.3% respectively. The results showed that our methods could get a small but stable improvements on English (p<.08).

Subcategory Refined from the Original Training Set

PN-0	**援外(aid foreign)**, *叔婶(aunt)*, *自家(self)*, *汝(you)*, **予(donate)**, 那些(those), 相貌(appearence), *自个儿(self)*, *咱们(we)*, 彼(that), 以上(the above), 那边(there), 其他(other), 以下(below)

Subcategories Fefined from the Good Starting Annotations

PN-0	*叔婶(aunt)*, *自家(self)*, *自个儿(self)*, *咱们(we)*, *汝(you)*
PN-Event-0	**援外(aid foreign)**, **予(donate)**
PN-AttributeValue-2	以上(the above), 那些(those), 彼(that), 其他(other), 以下(below)

Table 5: Several subcategories that generated from the original training set and the good starting annotations respectively.

Method	F_1
Baseline	90.1
Semantic Annotations	90.2
Semantic Annotations & Knowledge-based Criterion	**90.3**

Table 4: Performances at the fifth iteration on English (sentences of length 40 or less) with three methods: the baseline, the parser trained on the semantic annotations with automatic method, and the parser trained on the semantic annotations with knowledge-based criterion.

These results on English were preliminary, and we did not introduce any language dependent operation such as morphological processing. Since only the lemma of English words can be found in HowNet, we just annotated two kinds of POS tags "VB"(Verb, base form) and "NN"(Noun, singular or mass) with semantic knowledge, on the contrary, we annotated almost all POS tags whose corresponding words could be found in HowNet on Chinese. This might be the reason that the improvement on the English Treebank was much smaller than that of Chinese. It is expected to achieve more improvements through some morphological analysis in the future.

4.4 Results and Analysis

So far, a new strategy has been introduced to refine the grammars in two steps, and achieved significant improvements on parsing performance. In this section, we analyze the grammars learned at different steps, attempting to explain how the semantic knowledge works.

It is hard to inspect all the grammars by hand. Since the semantic knowledge is mainly used for generating and splitting new semantic-related tags in our method, we focus on the refined subcategories of these tags.

First, we examine the refined subcategories of POS tags, which are generated from the original training set and the good starting annotations respectively. Several subcategories are listed and compared in Table 5, along with their frequent words. It can be seen that the subcategories refined with semantic knowledge are more consistent than the previous one. For example, the subcategory "PN-0", which is refined from the original training set, produces a lot of words without semantic consistence. In contrast, we refine the subcategories "PN-0", "PN-Event-0" and "PN-AttributeValue-2" from the good starting annotations. Each of them produces a small but semantic consistent word set.

In order to inspect the difference between the automatic splitting process and the semantic based one, we compare the numbers of subcategories refined in these two processes. Since it is hard to list all the semantic-related tags here, three parts of the semantic-related tags were selected and listed in Table 6, along with the number of their subcategories. The first part is the noun and verb related tags, which are most heavily split in both two processes. It is clear that the semantic based splitting process can generate more subcategories than the automatic one, because the semantic structures of noun and verb are sophisticated. The second part lists the tags that have much more subcategories (≥ 4) from the automatic splitting process than the semantic based one, and the third part vice verse. It can be seen that most of the subcategories in the second part are functional categories, while most of the subcategories in the third part are content categories. It means that the semantic based splitting process is prone to generating less subcategory for the functional categories, but more subcategories for the content categories. This tendency is in accordance with the linguistic intuition. We believe that it is the main effect

Semantic-related tag	Automatic split number	Semantic based split numebr
NN-Attribute	30	30
NN-AttributeValue	25	**27**
NN-Entity	32	32
NN-Event	**31**	30
VV-Attribute	2	2
VV-AttributeValue	27	27
VV-Entity	22	**26**
VV-Event	29	**32**
BA-event	**13**	5
CS-AttributeValue	**29**	16
CS-entity	**22**	15
OD-Attribute	**13**	7
PN-Attribute	**26**	22
AS-AttributeValue	2	**7**
JJ-event	4	**8**
NR-AttributeValue	9	**13**
NT-event	12	**18**
VA-AttributeValue	22	**27**
VA-event	7	**11**

Table 6: The number of subcategories learned from two approaches: the automatic hierarchical state-splitting, and the semantic based splitting.

of our knowledge-based criterion, because it adjusts the splitting results dynamically with semantic knowledge, which can alleviate the overfitting risk.

5 Conclusions

In this paper, we present a novel approach to integrate semantic knowledge into the hierarchical state-split process for grammar refinement, which yields better accuracies on Chinese than previous methods. The improvements are mainly owing to two aspects. Firstly, the original treebank is initialized by annotating the tag of each POS node with the most general hypernym of its terminal word, which reduces the search space for the EM algorithm and brings an initial restrict to the following splitting step. Secondly, the splitting process is supervised by a knowledge-based criterion with the new semantic-related tags. Benefitting from the hierarchical semantic knowledge, the proposed approach alleviates the overfitting risk in a knowledge-driven manner. Experimental results reveal that the semantic knowledge is of great use to syntactic disambiguation. The further analysis on the refined grammars shows that, our method tends to split the content categories more often than the baseline method and the function classes less often.

Acknowledgments

We thank Yaozhong Zhang for the enlightening discussions. We also thank the anonymous reviewers who gave very helpful comments. The work was supported in part by the National Natural Science Foundation of China (60535030; 60605016), the National High Technology Research and Development Program of China (2006AA010103), the National Key Basic Research Program of China (2004CB318005, 2004CB318105).

References

E. Agirre, T. Baldwin and D. Martinez. 2008. Improving parsing and PP attachment performance with sense information. In *Proc. of ACL'08*, pages 317-325.

D. Bikel. 2000. A statistical model for parsing and word-sense disambiguation. In *Proc. of EMNLP/VLC'2000*, pages 155-163.

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

E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proc. of AAAI'97*, pages 598-603.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. of NAACL'00*, pages 132-139.

E Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxEnt discriminative reranking. In *Proc. of ACL'05*, pages 173-180.

D. Chiang and D. Bikel. 2002. Recovering latent information in treebanks. In *Proc. of COLING'02*, pages 183-189.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of ACL'97*, pages 16-23.

M. Collins. 1999. Head-driven statistical models for natural language parsing. *Ph.D. thesis*, U. of Pennsylvania.

Z. Dong and Q. Dong. 2000. HowNet Chinese-English conceptual database. Technical Report Online Software Database, Released at ACL. http://www.keenage.com.

S. Fujita, F. Bond, S. Oepen and T. Tanaka 2007. Exploiting semantic information for HPSG parse selection. *In ACL 2007 Workshop on Deep Linguistic Processing*, pages 25-32.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613-631.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL'03*, pages 423-430.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313-330.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proc. of ACL'05*, pages 75-82.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *Proc. of ARPA-HLT Workshop.*, pages 240-243.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. of COLING-ACL'06*, pages 443 – 440.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. of HLT-NAACL'07*, pages 404-411.

D. Xiong, S. Li, Q. Liu, S. Lin, and Y. Qian. 2005. Parsing the Penn Chinese treebank with semantic knowledge. In *Proc. of IJCNLP'05*, pages 70-81.

N. Xue, F.-D. Chiou, and M. Palmer. 2002. Building a large scale annotated Chinese corpus. In *Proc. of COLING'02*, pages 1-8.

Y. Zhang. 2008. The Study and Realization of Chinese Parsing with Semantic and Sentence Type Information. Master thesis, Peking University.

Bayesian Learning of Phrasal Tree-to-String Templates

Ding Liu and **Daniel Gildea**
Department of Computer Science
University of Rochester
Rochester, NY 14627
{dliu, gildea}@cs.rochester.edu

Abstract

We examine the problem of overcoming noisy word-level alignments when learning tree-to-string translation rules. Our approach introduces new rules, and re-estimates rule probabilities using EM. The major obstacles to this approach are the very reasons that word-alignments are used for rule extraction: the huge space of possible rules, as well as controlling overfitting. By carefully controlling which portions of the original alignments are re-analyzed, and by using Bayesian inference during re-analysis, we show significant improvement over the baseline rules extracted from word-level alignments.

1 Introduction

Non-parametric Bayesian methods have been successfully applied to directly learn phrase pairs from a bilingual corpus with little or no dependence on word alignments (Blunsom et al., 2008; DeNero et al., 2008). Because such approaches directly learn a generative model over phrase pairs, they are theoretically preferable to the standard heuristics for extracting the phrase pairs from the many-to-one word-level alignments produced by the IBM series models (Brown et al., 1993) or the Hidden Markov Model (HMM) (Vogel et al., 1996). We wish to apply this direct, Bayesian approach to learn better translation rules for syntax-based statistical MT (SSMT), by which we specifically refer to MT systems using Tree-to-String (TTS) translation templates derived from syntax trees (Liu et al., 2006; Huang et al., 2006; Galley et al., 2006; May and Knight, 2007), as opposed to *formally syntactic* systems such as Hiero (Chiang, 2007). The stumbling block preventing us from taking this approach is the extremely large space of possible TTS templates when no word alignments are given. Given a sentence pair and syntax tree over one side, there are an exponential number of potential TTS templates and a polynomial number of phrase pairs. In this paper, we explore methods for restricting the space of possible TTS templates under consideration, while still allowing good templates to emerge directly from the data as much as possible. We find an improvement in translation accuracy through, first, using constraints to limit the number of new templates, second, using Bayesian methods to limit which of these new templates are favored when re-analyzing the training data with EM, and, third, experimenting with different renormalization techniques for the EM re-analysis.

We introduce two constraints to limit the number of TTS templates that we extract directly from tree/string pairs without using word alignments. The first constraint is to limit direct TTS template extraction to the part of the corpus where word alignment tools such as GIZA++ do poorly. There is no reason not to re-use the good alignments from GIZA++, which holds a very competitive baseline performance. As already mentioned, the noisy alignments from GIZA++ are likely to cross the boundaries of the tree constituents, which leads to comparatively big TTS templates. We use this fact as a heuristic to roughly distinguish noisy from good word alignments.[1] Here we define *big templates* as those with more than 8 symbols in their right hand sides (RHSs). The word alignments in big templates are considered to be noisy and will be recomposed by extracting smaller TTS templates. Another reason to do extraction on big templates is that the applicability of big templates to new sentences is very limited due to their size, and the portion of the training data from which they are extracted is effectively wasted. The second constraint, after choosing the

[1] Precisely differentiating the noisy/good word alignments is as hard as correctly aligning the words.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1308–1317,
Singapore, 6-7 August 2009.

extraction site, is to extract the TTS templates all the way down to the leaves of the hosting templates. This constraint limits the number of possible left hand sides (LHSs) to be equal to the number of tree nodes in the hosting templates. The entire extraction process can be summarized in 3 steps:

1. Compute word alignments using GIZA++, and generate the basic TTS templates.

2. Select big templates from the basic TTS templates in step 1, and extract smaller TTS templates all the way down to the bottom from big templates, without considering the pre-computed word alignments.

3. Combine TTS templates from step 1 and step 2 and estimate their probabilities using Variational Bayes with a Dirichlet Process prior.

In step 2, since there are no constraints from the pre-computed word alignments, we have complete freedom in generating all possible TTS templates to overcome noisy word alignments. We use variational EM to approximate the inference of our Bayesian model and explore different normalization methods for the TTS templates. A two-stage normalization is proposed by combining LHS-based normalization with normalization based on the root of the LHS, and is shown to be the best model when used with variational EM.

Galley et al. (2006) recompose the TTS templates by inserting unaligned target words and combining small templates into bigger ones. The recomposed templates are then re-estimated using the EM algorithm described in Graehl and Knight (2004). This approach also generates TTS templates beyond the precomputed word alignments, but the freedom is only granted over unaligned target words, and most of the pre-computed word alignments remain unchanged. Other prior approaches towards improving TTS templates focus on improving the word alignment performance over the classic models such as IBM series models and Hidden Markov Model (HMM), which do not consider the syntactic structure of the aligning languages and produce syntax-violating alignments. DeNero and Klein (2007) use a syntax-based distance in an HMM word alignment model to favor syntax-friendly alignments. Fossum et al. (2008) start from the GIZA++ alignment and incrementally delete bad links based on a discriminative model with syntactic features. This approach can only find a better subset of the GIZA++ alignment and requires a parallel corpus with gold-standard word alignment for training the discriminative model. May and Knight (2007) factorize the word alignment into a set of re-orderings represented by the TTS templates and build a hierarchical syntax-based word alignment model. The problem is that the TTS templates are generated by the word alignments from GIZA++, which limits the potential of the syntactic re-alignment. As shown by these prior approaches, directly improving the word alignment either falls into the framework of many-to-one alignment, or is substantially confined by the word alignment it builds upon.

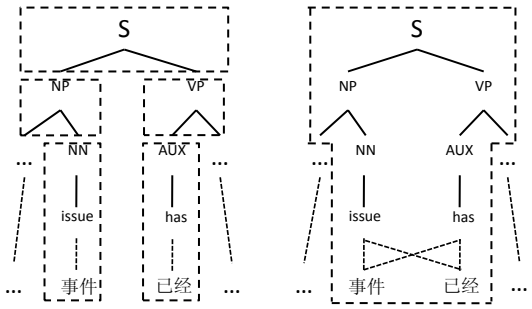


Figure 1: 5 small TTS templates are extracted based on the correct word alignments (left), but only 1 big TTS template (right) can be extracted when the cross-boundary noisy alignments are added in.

The remainder of the paper focuses on the Bayesian approach to learning TTS templates and is organized as follows: Section 2 describes the procedure for generating the candidate TTS templates; Section 3 describes the inference methods used to learn the TTS templates; Section 4 gives the empirical results, Section 5 discusses the characteristics of the learned TTS templates, and Section 6 presents the conclusion.

2 Extracting Phrasal TTS Templates

The Tree-to-String (TTS) template, the most important component of a SSMT system, usually contains three parts: a fragment of a syntax tree in its left hand side (LHS), a sequence of words and variables in its right hand side (RHS), and a probability indicating how likely the template is to be used in translation. The RHS of a TTS template shows one possible translation and reordering of its LHS. The variables in a TTS template are further transformed using other TTS templates, and the recursive process continues until there are no variables left. There are two ways

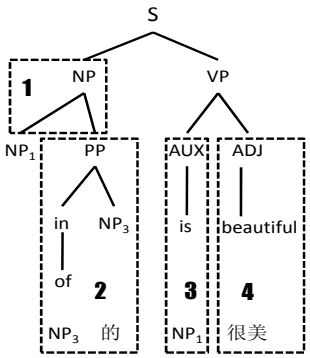


Figure 2: Examples of valid and invalid templates extracted from a big Template. Template 1, invalid, doesn't go all the way down to the bottom. Template 2 is valid. Template 3, invalid, doesn't have the same set of variables in its LHS/RHS. Template 4, invalid, is not a phrasal TTS template.

that TTS templates are commonly used in machine translation. The first is *synchronous parsing* (Galley et al., 2006; May and Knight, 2007), where TTS templates are used to construct synchronous parse trees for an input sentence, and the translations will be generated once the synchronous trees are built up. The other way is the TTS *transducer* (Liu et al., 2006; Huang et al., 2006), where TTS templates are used just as their name indicates: to transform a source parse tree (or forest) into the proper target string. Since synchronous parsing considers all possible synchronous parse trees of the source sentence, it is less constrained than TTS transducers and hence requires more computational power. In this paper, we use a TTS transducer to test the performance of different TTS templates, but our techniques could also be applied to SSMT systems based on synchronous parsing.

2.1 Baseline Approach: TTS Templates Obeying Word Alignment

TTS templates are commonly generated by decomposing a pair of aligned source syntax tree and target string into smaller pairs of tree fragments and target string (i.e., the TTS templates). To keep the number of TTS templates to a manageable scale, only the non-decomposable TTS templates are generated. This algorithm is referred to as GHKM (Galley et al., 2004) and is widely used in SSMT systems (Galley et al., 2006; Liu et al., 2006; Huang et al., 2006). The word alignment used in GHKM is usually computed independent of the syntactic structure, and as DeNero and Klein (2007) and May and Knight (2007) have noted,

Ch-En	En-Ch	Union	Heuristic
28.6%	33.0%	45.9%	20.1%

Table 1: Percentage of corpus used to generate big templates, based on different word alignments

	9-12	13-20	≥21
Ch-En	18.2%	17.4%	64.4%
En-Ch	15.9%	20.7%	63.4%
Union	9.8%	15.1%	75.1%
Heuristic	24.6%	27.9%	47.5%

Table 2: In the selected big templates, the distribution of words in the templates of different sizes, which are measured based on the number of symbols in their RHSs

is not the best for SSMT systems. In fact, noisy word alignments cause more damage to a SSMT system than to a phrase based SMT system, because the TTS templates can only be derived from tree constituents. If some noisy alignments happen to cross over the boundaries of two constituents, as shown in Figure 2, a much bigger tree fragment will be extracted as a TTS template. Even though the big TTS templates still carry the original alignment information, they have much less chance of getting matched beyond the syntax tree where they were extracted, as we show in Section 4. In other words, a few cross-boundary noisy alignments could disable a big portion of a training syntax tree, while for a phrase-based SMT system, their effect is limited to the phrases they align. As a rough measure of how the training corpus is affected by the big templates, we calculated the distribution of target words in big and non-big TTS templates. The word alignment is computed using GIZA++[2] for the selected 73,597 sentence pairs in the FBIS corpus in both directions and then combined using union and heuristic diagonal growing (Koehn et al., 2003). Table 1 shows that big templates consume 20.1% to 45.9% of the training corpus depending on different types of word alignments. The statistics indicate that a significant portion of the training corpus is simply wasted, if the TTS templates are extracted based on word alignments from GIZA++. On the other hand, it shows the potential for improving an SSMT system if we can efficiently re-use the wasted training corpus. By further examining the selected big templates, we find that the most common form of big templates is a big skeleton template starting

[2]GIZA++ is available at http://www.fjoch.com/GIZA++.html

from the root of the source syntax tree, and having many terminals (words) misaligned in the bottom. Table 2 shows, in the selected big templates, the distribution of words in the templates of different sizes (measured based on the number of symbols in their RHS). We can see that based on either type of word alignment, the most common big templates are the TTS templates with more than 20 symbols in their RHSs, which are generally the big skeleton templates. The advantage of such big skeleton templates is that they usually have good marginal accuracy[3] and allow accurate smaller TTS templates to emerge.

2.2 Liberating Phrasal TTS Templates From Noisy Word Alignments

To generate better TTS templates, we use a more direct way than modifying the underlying word alignment: extract smaller phrasal TTS templates from the big templates without looking at their pre-computed word alignments. We define *phrasal* TTS templates as those with more than one symbol (word or non-terminal) in their LHS. The reason to consider only phrasal TTS templates is that they are more robust than the word-level TTS templates in addressing the complicated word alignments involved in big templates, which are usually not the simple type of one-to-many or many-to-one. Abandoning the pre-computed word alignments in big templates, an extracted smaller TTS template can have many possible RHSs, as long as the two sides have the same set of variables. Note that the freedom is only given to the alignments of the words; for the variables in the big templates, we respect the pre-computed word alignments. To keep the extracted smaller TTS templates to a manageable scale, the following two constraints are applied:

1. The LHS of extracted TTS templates should go all the way down to the bottom of the LHS of the big templates. This constraint ensures that at most N LHSs can be extracted from one big Template, where N is the number of tree nodes in the big Template's LHS.

2. The number of leaves (including both words and variables) in an extracted TTS template's LHS should not exceed 6. This constraint limits the size of the extracted TTS templates.

[3]Here, marginal accuracy means the correctness of the TTS template's RHS corresponding to its LHS.

(VP (AUX is) (ADJ beautiful)) ⟶ 很美

(PP (IN of) NP_3) ⟶ NP_3 的

(NP NP_1 (PP (IN of) NP_3)) ⟶ NP_3 的 NP_1

(NP NP_1 (PP (IN of) NP_3)) ⟶ NP_3 的 NP_1 很美

Figure 3: All valid templates that can be extracted from the example in Figure 2.1

```
for all template t do
    if size(t.rhs) > 8 then
        for all tree node s in t.lhs do
            subt = subtree(s, t.lhs);
            if leaf_num(subt) ≤ 6 then
                for i=1:size(t.rhs) do
                    for j=i:size(t.rhs) do
                        if valid(subt, i, j) then
                            create_template(subt, i, j);
```

Figure 4: Algorithm that liberates smaller TTS Templates from big templates

As we show in Section 4, use of bigger TTS templates brings very limited performance gain.

Figure 2.2 describes the template liberating algorithm running in $O(NM^2)$, where N denotes the number of tree nodes in the LHS of the input big Template and M denotes the length of the RHS. In the algorithm, function *valid* returns *true* if there are the same set of variables in the left/right hand side of an extracted TTS template; *subtree(x, y)* denotes the sub-tree in *y* which is rooted at *x* and goes all the way down to *y*'s bottom. Figure 2.1 shows valid and invalid TTS templates which can be extracted from an example hosting TTS template. Note that, in order to keep the example simple, the hosting TTS template only has 4 symbols in its RHS, which does not qualify as a big template according to our definition. Figure 2.2 shows the complete set of valid TTS templates which can be extracted from the example TTS template. The subscripts of the non-terminals are used to differentiate identical non-terminals in different positions. The extraction process blindly releases smaller TTS templates from the big templates, among which only a small fraction are correct TTS templates. Therefore, we need an inference method to raise the weight of the correct templates and decrease the weight of the noisy templates.

3 Estimating TTS Template Probability

The Expectation-Maximization (EM) algorithm (Dempster et al., 1977) can be used to estimate the TTS templates' probabilities, given a generative model addressing how a pair of source syntax tree and target string is generated. There are two commonly used generative models for syntax-based MT systems, each of which corresponds to a normalization method for the TTS templates. The LHS-based normalization (LHSN) (Liu et al., 2006; Huang et al., 2006), corresponds to the generative process where the source syntax subtree is first generated, and then the target string is generated given the source syntax subtree. The other one is normalization based on the root of the LHS (ROOTN) (Galley et al., 2006), corresponding to the generative process where, given the root of the syntax subtree, the LHS syntax subtree and the RHS string are generated simultaneously. By omitting the decomposition probability in the LHS-based generative model, the two generative models share the same formula for computing the probability of a training instance:

$$Pr(T,S) = \sum_{R} Pr(T,S,R) = \sum_{R} \left(\prod_{t \in R} Pr(t) \right)$$

where T and S denote the source syntax tree and target string respectively, R denotes the decomposition of (T,S), and t denotes the TTS template. The expected counts of the TTS templates can then be efficiently computed using an inside-outside-like dynamic programming algorithm (May and Knight, 2007).

LHSN, as shown by Galley et al. (2006), cannot accurately restore the true conditional probabilities of the target sentences given the source sentences in the training corpus. This indicates that LHSN is not good at predicting unseen sentences or at translating new sentences. But this deficiency does not affect its ability to estimate the expected counts of the TTS templates, because the posteriors of the TTS templates only depend on the comparative probabilities of the different derivations of a training instance (a pair of tree and string). In fact, as we show in Section 4, LHSN is better than ROOTN in liberating smaller TTS templates out of the big templates, since it is less biased to the big templates in the EM training.[4] Because the two normalization methods have their own strength and weakness, both of them are used in our EM algorithm: LHSN is used in all EM iterations except the last one to compute the expected counts of the TTS templates, and ROOTN is used in the last EM iteration to compute the final probabilities of the TTS templates. This two-stage normalization method is denoted as MIXN in this paper.

E-step:
for all pair of syntax tree T and target string S **do**
  for all TTS Template t **do**
    $EC(t) += \frac{\sum_{R:t \in R} Pr(T,S,R)^{\beta}}{\sum_{R'} Pr(T,S,R')^{\beta}}$;
Increase β;

M-step:
for all TTS Template t **do**
  if it is the last iteration **then**
    $Pr(t) = \frac{EC(t)}{\sum_{t':t'.root=t.root} EC(t')}$;
  else
    $Pr(t) = \frac{EC(t)}{\sum_{t':t'.lhs=t.lhs} EC(t')}$;

Figure 5: EM Algorithm For Estimating TTS Templates

Deterministic Annealing (Rose et al., 1992) is is used in our system to speed up the training process, similar to Goldwater et al. (2006). We start from a high temperature and gradually decrease the temperature to 1; we find that the initial high temperature can also help small templates to survive the initial iterations. The complete EM framework is sketched in Figure 3, where β is the inverse of the specified temperature, and *EC* denotes the expected count.

3.1 Bayesian Inference with the Dirichlet Process Prior

Bayesian inference plus the Dirichlet Process (DP) have been shown to effectively prevent MT models from overfitting the training data (DeNero et al., 2008; Blunsom et al., 2008). A similar approach can be applied here for SSMT by considering each TTS template as a cluster, and using DP to adjust the number of TTS templates according to the training data. Note that even though there is a size limitation on the liberated phrasal TTS templates, standard EM will still tend to overfit the training data by pushing up the probabilities of the big templates from the noisy word alignments. The complete generative process, integrating the DP prior and the generative models described in

[4] Based on LHSN, the difference between the probability of a big Template and the product of the probabilities of its decomposing TTS templates is much less than the one based on ROOTN, thus LHSN gives comparably more expected counts to the smaller TTS templates than ROOTN.

for all TTS Template t **do**
  if it is the last iteration **then**
    $Pr(t) = \frac{\exp(\psi(EC(t)+\alpha G_0(t)))}{\exp\left(\psi\left(\left(\sum_{t':t'.root=t.root} EC(t')\right)+\alpha\right)\right)}$;
  else
    $Pr(t) = \frac{\exp(\psi(EC(t)+\alpha G_0(t)))}{\exp\left(\psi\left(\left(\sum_{t':t'.lhs=t.lhs} EC(t')\right)+\alpha\right)\right)}$;

Figure 6: M-step of the Variational EM

Section 3.1, is given below:

$$\begin{aligned} \theta^r \mid \{\alpha_r, G_0^r\} &\sim DP(\alpha_r, G_0^r) \\ t \mid \theta^{t.root} &\sim \theta^{t.root} \\ (T,S) \mid \{SG, \{t\}, \theta\} &\sim SG(\{t\}, \theta) \end{aligned}$$

where G_0 is a base distribution of the TTS templates, t denotes a TTS template, $\theta^{t.root}$ denotes the multinomial distribution over TTS templates with the same root as t, SG denotes the generative model for a pair of tree and string in Section 3.1, and α is a free parameter which adjusts the rate at which new TTS templates are generated.

It is intractable to do exact inference under the Bayesian framework, even with a conjugate prior such as DP. Two methods are commonly used for approximate inference: Markov chain Monte Carlo (MCMC) (DeNero et al., 2008), and Variational Bayesian (VB) inference (Blunsom et al., 2008). In this paper, the latter approach is used because it requires less running time. The E-step of VB is exactly the same as standard EM, and in the M-step the digamma function ψ and the base distribution G_0 are used to increase the uncertainty of the model. Similar to standard EM, both LHS- and root-based normalizations are used in the M-step, as shown in Figure 3.1. For the TTS templates, which are also pairs of subtrees and strings, a natural choice of G_0 is the generative models described in Section 3.1. Because G_0 estimates the probability of the new TTS templates, the root-based generative model is superior to the LHS-based generative model and used in our approach.

3.2 Initialization

Since the EM algorithm only converges to a local minimum, proper initializations are needed to achieve good performance for both standard EM and variational EM. For the baseline templates derived from word alignments, the initial counts are set to the raw counts in the training corpus. For the templates blindly extracted from big templates, the raw count of a LHS tree fragment is distributed among their RHSs based on the likelihood of the template, computed by combining

for all big template t **do**
  for all template g extracted from t **do**
    $g.count$ = $g.lhs.count$ = 0;
  for all template g extracted from t **do**
    $g.count$ += w_in(g)×w_out(g, t);
    $g.lhs.count$ += w_in(g)×w_out(g, t);
  for all template g extracted from t **do**
    $g.init$ += $\frac{g.count}{g.lhs.count}$;

Figure 7: Compute the initial counts of the liberated TTS templates

the word-based inside/outside scores. The algorithm is sketched in Figure 3.2, where the inside score *w_in(g)* is the product of the IBM Model 1 scores in both directions, computed based on the words in g's LHS and RHS. The outside score *w_out(g, t)* is computed similarly, except that the IBM Model 1 scores are computed based on the words in the hosting template t's LHS/RHS excluding the words in g's LHS/RHS. The initial probabilities of the TTS templates are then computed by normalizing their initial counts using LHSN or ROOTN.

4 Experiments

We train an English-to-Chinese translation system using the FBIS corpus, where 73,597 sentence pairs are selected as the training data, and 500 sentence pairs with no more than 25 words on the Chinese side are selected for both the development and test data.[5] Charniak (2000)'s parser, trained on the Penn Treebank, is used to generate the English syntax trees. Modified Kneser-Ney trigram models are trained using SRILM (Stolcke, 2002) upon the Chinese portion of the training data. The trigram language model, as well as the TTS templates generated based on different methods, are used in the TTS transducer. The model weights of the transducer are tuned based on the development set using a grid-based line search, and the translation results are evaluated based on a single Chinese reference[6] using BLEU-4 (Papineni et al., 2002). Huang et al. (2006) used character-based BLEU as a way of normalizing inconsistent Chinese word segmentation, but we avoid this problem as the training, development, and test data are from the same source.

[5]The total 74,597 sentence pairs used in experiments are those in the FBIS corpus whose English part can be parsed using Charniak (2000)'s parser.

[6]BLEU-4 scores based on a single reference are much lower than the ones based on multiple references.

	E2C	C2E	Union	Heuristic
w/ Big	13.37	12.66	**14.55**	14.28
w/o Big	13.20	12.62	14.53	14.21

Table 3: BLEU-4 scores (test set) of systems based on GIZA++ word alignments

	≤ 5	≤ 6	≤ 7	≤ 8	$\leq \infty$
BLEU-4	14.27	14.42	14.43	14.45	14.55

Table 4: BLEU-4 scores (test set) of the *union* alignment, using TTS templates up to a certain size, in terms of the number of leaves in their LHSs

4.1 Baseline Systems

GHKM (Galley et al., 2004) is used to generate the baseline TTS templates based on the word alignments computed using GIZA++ and different combination methods, including *union* and the diagonal growing heuristic (Koehn et al., 2003). We also tried combining alignments from GIZA++ based on intersection, but it is worse than both single-direction alignments, due to its low coverage of training corpus and the incomplete translations it generates. The baseline translation results based on ROOTN are shown in Table 4.1. The first two columns in the table show the results of the two single direction alignments. *e2c* and *c2e* denote the many English words to one Chinese word alignment and the many Chinese words to one English word alignment, respectively. The two rows show the results with and without the big templates, from which we can see that removing the big templates does not affect performance much; this verifies our postulate that the big templates have very little chance of being used in the translation. Table 4.1, using the *union* alignments as the representative and measuring a template's size by the number of leaves in its LHS, also demonstrates that using big TTS templates brings very limited performance gain.

The result that the union-based combination outperforms either single direction alignments and even the heuristic-based combination, combined with the statistics of the disabled corpus in Section 2.2, shows that more disabled training corpus actually leads to better performance. This can be explained by the fact that the *union* alignments have the largest number of noisy alignments gathered together in the big templates, and thus have the least amount of noisy alignments which lead to small and low-quality TTS templates.

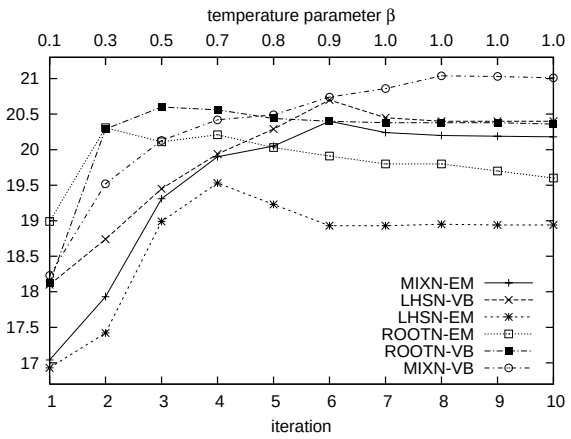


Figure 8: BLEU-4 scores (development set) of annealing EM and annealing VB in each iteration.

4.2 Learning Phrasal TTS Templates

To test our learning methods, we start with the TTS templates generated based on *e2c*, *c2e*, and *union* alignments using GHKM. This gives us 0.98M baseline templates. We use the big templates from the *union* alignments as the basis and extract 10.92M new phrasal TTS templates, which, for convenience, are denoted by NEW-PHR. Because based on Table 1 and Table 2 the *union* alignment has the greatest number of alignment links and therefore produces the largest rules, this gives us the greatest flexibility in realigning the input sentences. The baseline TTS templates as well as NEW-PHR are initialized using the method in Section 3.3 for both annealing EM and annealing VB. To simplify the experiments, the same Dirichlet Process prior is used for all multinomial distributions of the TTS templates with different roots. G_0 in the Dirichlet prior is computed based on the 1-level TTS templates selected from the baseline TTS templates, so that the big templates are efficiently penalized. The training algorithms follow the same annealing schedule, where the temperature parameter β is initialized to 0.1, and gradually increased to 1.

We experiment with the two training algorithms, annealing EM and annealing VB, with different normalization methods. The experimental results based on the development data are shown in Figure 4.2, where the free parameter α of annealing VB is set to 1, 100, and 100 respectively for ROOTN, LHSN, and MIXN. The results verify that LHSN is worse than ROOTN in predicting the translations, since MIXN outperforms LHSN with both annealing EM and VB. ROOTN is on par with MIXN and much better

	Max Likelihood		Annealing EM		Annealing VB	
	w/o new-phr	with new-phr	w/o new-phr	with new-phr	w/o new-phr	with new-phr
LHSN	14.05	13.16	14.31	15.33	14.82	16.15
ROOTN	14.50	13.49	14.90	16.06	14.76	16.12
MIXN	NA	NA	14.82	16.37	14.93	**16.84**

Table 5: BLEU-4 scores (test set) of different systems.

	Initial Template		Final Template	
	number	new-phr%	number	new-phr%
ROOTN	11.9M	91.8%	408.0K	21.9%
LHSN	11.9M	91.8%	557.2K	29.8%
MIXN	11.9M	91.8%	500.5K	27.6%

Table 6: The total number of templates and the percentage of NEW-PHR, in the beginning and end of annealing VB

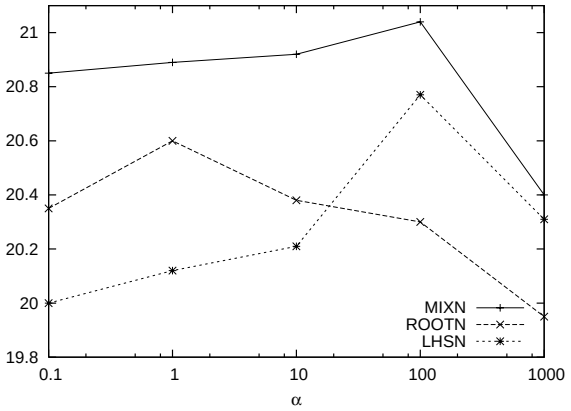


Figure 9: BLEU-4 scores (development set) of annealing VB with different α.

than LHSN when annealing EM is used; but with annealing VB, it is outperformed by MIXN by a large margin and is even slightly worse than LHSN. This indicates that ROOTN is not giving large expected counts to NEW-PHR and leaves very little space for VB to further improve the results. For all the normalization methods, annealing VB outperforms annealing EM and maintains a longer ascending path, showing better control of overfitting for the Bayesian models. Figure 4.2 shows the optimized results of the development set based on annealing VB with different α. The best performance is achieved as α approaches 1, 100, and 100 for ROOTN, LHSN and MIXN respectively. The α parameter can be viewed as a weight used to balance the expected counts and the probabilities from G_0. Thus it is reasonable for LHSN and MIXN to have bigger optimal α than ROOTN, since ROOTN gives lower expected counts to NEW-PHR than LHSN and MIXN do.

To see the contribution of the phrasal template extraction in the performance gain, MT experiments are conducted by turning this component on and off. Results on the test set, obtained by using parameters optimized on the development set, are shown in Table 4.2. The template counts used in the Max-Likelihood training are the same as the ones used in the initialization of annealing EM and VB. Results show that for annealing EM and VB, use of NEW-PHR greatly improves performance, while for the Max-Likelihood training, use of NEW-PHR hurts performance. This is not surprising, because Max-Likelihood training cannot efficiently filter out the noisy phrasal templates introduced in the initial NEW-PHR. Another observation is that annealing VB does not always outperform annealing EM. With NEW-PHR turned on, annealing VB shows consistent superiority over annealing EM; while without NEW-PHR, it only outperforms annealing EM based on LHSN and MIXN, and the improvement is not as big as when NEW-PHR is turned on. This indicates that without NEW-PHR, there is less need to use VB to shrink down the size of the template set. Table 4.2 shows the statistics of the initial template set including NEW-PHR and the final TTS template set after annealing VB is conducted, where we can see annealing VB efficiently reduces NEW-PHR to a relatively small size and results in much more compact systems than the system based on the baseline templates from GIZA++ alignments. Comparing with the best GIZA++-based system *union*, our best system, utilizing NEW-PHR and the two-stage template normalization, demonstrates the strength of annealing VB by an absolute improvement of 2.29% in BLEU-4 score, from 14.55 to 16.84. This improvement is significant at $p < 0.005$ based on 2000 iterations of paired bootstrap re-sampling of the test set (Koehn, 2004).

5 Discussion

Our experimental results are obtained based on a relatively small training corpus, the improved performance may be questionable when a larger training corpus is used. Someone may wonder if the performance gain primarily comes from the

Many-to-one Alignment	
(VP (VB make) (NP (DT a) (JJ complete) (NN statement)))	充分陈述
(S (VP VBG (NP (DT the) (NN mass) (NN line)) PP))	PP VBG 群众路线
(PP (TO to) (NP (DT the) (JJS greatest) (NN extent)))	最大限度 地
(PP (IN of) (NP (JJ peaceful) (NNP coexistence)))	和平共处
Many-to-many Alignment	
(VP (VBN based) (PP (IN on) (NP (JJ actual) (NNS needs))))	从 实际 出发
(PP (IN into) (NP (NP (DT the) (NNS hands)) PP))	掌握 在 PP 手上
(VP (VBP exercise) (NP (JJ strict) (NN self-discipline)))	严 以 律 己
(SBAR (S (NP (DT the) (VBG aging) NN) (VP (aux is) NP)))	NN 老 龄 化 是 NP
(NP NP_1 PP (, ,) (VP (VBN centered) (PP (IN around) NP_2)))	以 NP_2 为 核心 的 NP_1 PP
Allowance of Bad Word Segmentation	
(NP (NP (NNP japan) (POS 's)) (NNP sdf) (NNP navy))	日本海 上 自卫队
(NP (PDT all) (NP (NNS people) (POS 's)) (NNS organizations))	各人 民团 体

Figure 10: Examples of the learned TTS templates

reduced out of vocabulary (OOV) ratio. We examined the OOV ratio of the test set with/without the learned TTS templates, and found the difference was very small. In fact, our method is designed to learn the phrasal TTS templates, and explictly avoids lexical pairs. To further understand the characteristics of the learned TTS templates, we list some representative templates in Figure 4.2 classified in 3 groups. The group *Many-to-one Alignment* and *Many-to-many Alignment* show the TTS templates based on complicated word alignments, which are difficult to compute based on the existing word alignment models. These templates do not have rare English words, whose translation cannot be found outside the big templates. The difficulty lies in the non-literal translation of the source words, which are unlikely to learnt by soly increasing the size of the training corpus. One other interesting observation is that our learning method is tolerant to noisy Chinese word segmentation, as shown in group *Allowance of Bad Word Segmentation*.

6 Conclusion

This paper proposes a Bayesian model for extracting the Tree-to-String templates directly from the data. By limiting the extraction to the big templates from the pre-computed word alignments and applying a set of constraints, we restrict the space of possible TTS templates under consideration, while still allowing new and more accurate templates to emerge from the training data. The empirical results demonstrate the strength of our approach, which outperforms the GIZA++-based systems by a large margin. This encourages a move from word-alignment-based systems to systems based on consistent, end-to-end probabilistic modeling. Because our Bayesian model employs a very simple prior, more sophisticated generative models provide a possible direction for further experimentation.

Acknowledgments This work was supported by NSF grants IIS-0546554 and ITR-0428020.

References

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. Bayesian synchronous grammar induction. In *Neural Information Processing Systems (NIPS)*.

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

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL-01*, pages 132–139.

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

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society*, 39(1):1–21.

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *Proceedings of ACL-07*, pages 17–24.

John DeNero, Alexandre Bouchard-Cote, and Dan Klein. 2008. Sampling alignment structure under a bayesian translation model. In *EMNLP08*.

Victoria Fossum, Kevin Knight, and Steven Abney. 2008. Using syntax to improveword alignment precision for syntax-based machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, Columbus, Ohio. ACL.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of NAACL-04*, pages 273–280.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING/ACL-06*, pages 961–968, July.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proceedings of NAACL-04*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, MA.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL-03*, Edmonton, Alberta.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395, Barcelona, Spain, July.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING/ACL-06*, Sydney, Australia, July.

J. May and K. Knight. 2007. Syntactic re-alignment models for machine translation. In *Proceedings of EMNLP*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL-02*.

K. Rose, E. Gurewitz, and G. C. Fox. 1992. Vector quantization by deterministic annealing. *IEEE Transactions on Information Theory*, 38(4):1249–1257, July.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *COLING-96*, pages 836–841.

Human-competitive tagging using automatic keyphrase extraction

Olena Medelyan, Eibe Frank, Ian H. Witten
Computer Science Department
University of Waikato
{olena,eibe,ihw}@cs.waikato.ac.nz

Abstract

This paper connects two research areas: automatic tagging on the web and statistical keyphrase extraction. First, we analyze the quality of tags in a collaboratively created folksonomy using traditional evaluation techniques. Next, we demonstrate how documents can be tagged automatically with a state-of-the-art keyphrase extraction algorithm, and further improve performance in this new domain using a new algorithm, "Maui", that utilizes semantic information extracted from Wikipedia. Maui outperforms existing approaches and extracts tags that are competitive with those assigned by the best performing human taggers.

1 Introduction

Tagging is the process of labeling web resources based on their content. Each label, or *tag*, corresponds to a topic in a given document. Unlike metadata assigned by authors, or by professional indexers in libraries, tags are assigned by end-users for organizing and sharing information that is of interest to them. The organic system of tags assigned by all users of a given web platform is called a *folksonomy*.

In contrast to traditional taxonomies painstakingly constructed by experts, a user can add any tags to a folksonomy. This leads to the greatest downside of tagging, inconsistency, which originates in the synonymy and polysemy of human language, as well as in the varying degrees of specificity used by taggers (Golder and Huberman, 2006). In traditional libraries, *consistency* is the primary evaluation criterion of indexing (Rolling, 1981). Much work has been done on describing the statistical properties of folksonomies, such as tag distribution and co-occurrences (Halpin *et al.*, 2007; Sigurbjörnsson *et al.*, 2008; Sood *et al.*, 2007), but to our knowledge there has been none on assessing the actual quality of tags. How well do human taggers perform? How consistent are they with each other?

One potential solution to inconsistency in folksonomies is to use suggestion tools that automatically compute tags for new documents (e.g. Mishne, 2006; Sood *et al.*, 2007; Heymann *et al.*, 2008). Interestingly, the blooming research on automatic tagging has so far not been connected to work on keyphrase extraction (e.g. Frank *et al.*, 1999; Turney, 2003; Hulth, 2004), which can be used as a tool for the same task (note: we use *tag* and *keyphrase* as synonyms). Instead of simple heuristics based on term frequencies and co-occurrence of tags, keyphrase extraction methods apply machine learning to determine typical distributions of properties common to manually assigned phrases, and can include analysis of semantic relations between candidate tags (Turney, 2003). How well do state-of-the-art keyphrase extraction systems perform compared to simple tagging techniques? How consistent are they with human taggers? These are questions we address in this paper.

Until now, keyphrase extraction methods have primarily been evaluated using a single set of keyphrases for each document, thereby largely ignoring the subjective nature of the task. Collaboratively tagged documents, on the other hand, offer multiple tag assignments by independent users, a unique basis for evaluation that we capitalize upon in this paper.

The experiments reported in this paper fill these gaps in the research on automatic tagging and keyphrase extraction. First, we analyze tagging consistency on the *CiteULike.org* platform for organizing academic citations. Methods traditionally used for the evaluation of professional indexing will provide insight into the quality of this folksonomy. Next, we extract a high quality corpus from CiteULike, containing documents that have been tagged consistently by the best human taggers.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1318–1327,
Singapore, 6-7 August 2009.

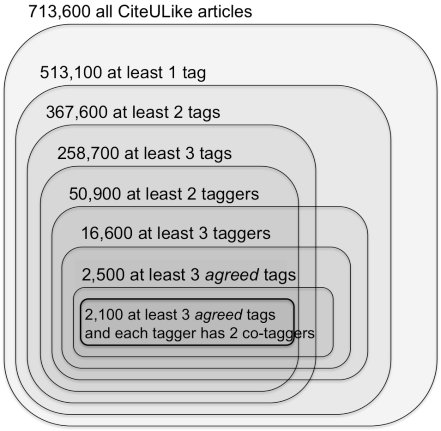


Figure 1. Quality control of CiteULike data

Following that, our goal is to build a system that matches the performance of these taggers. We first apply an existing approach proposed by Brooks and Montanez (2006) and compare it to the keyphrase extraction algorithm Kea (Frank *et al.*, 1999). Next we create a new algorithm, called Maui, that enhances Kea's successful machine learning framework with semantic knowledge retrieved from Wikipedia, new features, and a new classification model. We evaluate Maui using tag sets assigned to the same documents by several users and show that it is as consistent with CiteULike users as they are with each other.

Most of the computation required for automatic tagging with this method can be performed offline. In practice, it can be used as a tag suggestion tool that provides users with tags describing the main topics of newly added documents, which can then be corrected or enhanced by personal tags if required. This will improve consistency in the folksonomy without compromising its flexibility.

2 Collaboratively-tagged Data

CiteULike.org is a bookmarking service that resembles the popular *del.icio.us*, but concentrates on scholarly papers. Rather than replicating the full text of tagged papers it simply points to them on the web (e.g. PubMed, CiteSeer, ScienceDirect, Amazon) or in journals (e.g. HighWire, Nature). This avoids violating copyright but means that the full text of articles is not necessarily available. When entering new resources, users are encouraged to assign tags describing their content or reflecting their own grouping of the information. However, the system does not suggest tags. Moreover, users do not see other users' tags and are thus not biased in their tag choices.

2.1 Extracting a high quality tagged corpus

The CiteULike data set is freely available and contains information about which documents were tagged with what tags by which users (although identities are not provided).

CiteULike's 22,300 users have tagged 713,600 documents with 2.4M "tag assignments"— single applications of a tag by a user to a document. The two most popular tags, *bibtex-import* and *no-tag*, indicate an information source and a missing tag respectively. Most of the remainder describe particular concepts relevant to the documents. We exclude non-content tags from our experiments, e.g. personal tags like *to-read* or *todo*. Note that spam entries have been eliminated from the data set.

Because CiteULike taggers are not professional indexers, high quality of the assigned topics cannot be guaranteed. In fact, manual assessment of users' tags by human evaluators shows precision of 59% (Mishne, 2006) and 49% (Sood *et al.*, 2006). However, why is the opinion of human evaluators valued more than the opinion of taggers? We propose an alternative way of determining ground truth using an automatic approach to determine reliable tags: We concentrate on a subset of CiteULike containing documents that have been indexed with at least three tags on which at least two users have agreed.

In order to be able to measure the tagging consistency between the users, and then compare it to the algorithm's consistency, we need taggers who have tagged documents that some others had tagged. We say that two users are "co-taggers" if they have both tagged at least one common document. As well as restricting the document set, we only include taggers who have at least two co-taggers.

Figure 1 shows the proportions of CiteULike documents that are discarded in order to produce our high quality data set. The final set contains only 2,100 documents (0.3% of the original). Unfortunately, many of these are unavailable for download—for example, books at Amazon.com and ArXiv.org references cannot be crawled. We further restrict attention to two sources: HighWire and Nature, both of which provide easily-accessible PDFs of the full text.

The result is a set of 180 documents indexed by 332 taggers. A total of 4,638 tags were assigned by all taggers to documents in this set; however, the number of tags on which at least two users agreed is significantly smaller, namely 946. Still, this results in accurate tag sets that contain an average of five tags per document.

tagger	co-taggers	documents	consistency
1	1	5	71.4
2	1	5	71.4
3	6	5	57.9
4	6	6	51.0
5	11	12	50.4
6	2	5	50.1
7	4	6	48.3
8	8	8	47.1
9	13	16	45.4
10	12	8	44.4
11	7	6	43.5
12	7	6	41.7
13	8	5	40.9
14	7	6	39.7
15	9	13	38.8
16	4	5	38.4
17	12	9	37.3
18	4	14	36.1
19	9	8	35.9
20	10	11	33.7
21	7	6	33.1
22	6	5	33.0
23	7	10	32.1
24	11	16	31.7
25	8	13	30.6
26	6	8	30.6
27	9	6	29.8
28	10	12	29.0
29	8	6	28.8
30	9	10	27.9
31	10	8	26.7
32	8	7	26.3
33	10	5	25.6
34	8	7	21.0
35	9	9	18.3
36	3	6	7.9
average	**7.5**	**8.1**	**37.7**

Table 1. Consistency of the most prolific and most consistent taggers

Note that traditionally much smaller data sets are used to assess consistency of human indexers, because such sets need to be created specifically for the experiment. Collaborative tagging platforms like CiteULike can be mined for large collections of this kind in natural settings.

Most documents in the extracted set relate to the area of bioinformatics. To give an example, a document entitled *Initial sequencing and comparative analysis of the mouse genome* was tagged by eight users with a total of 22 tags. Four of them agreed on the tag *mouse*, but one used the broader term *rodents*. Three agreed on the tag *genome*, but one added *genome paper*, and another used the more specific *comparative genomics*. There are also cases when tags are written together, e.g. *genomepaper*, or with a prefix *key genome*, or in a different grammatical form: *sequence* vs. *sequencing*. This example shows that many inconsistencies in tags are not caused by personalized tag choices as Chirita *et al.* (2007) suggest, but rather stem from the lack of guidelines and uniform tag suggestions that a bookmarking service could provide.

2.2 Measuring tagging consistency

Traditional indexers aim for consistency, on the basis that this will enhance document retrieval (Leonard, 1975). Consistency is measured using experiments in which several people index the same documents—usually a small set of a few dozen documents. It is computed for pairs of indexers, by formulae such as Rolling's (1981):

$$Consistency(I_1, I_2) = \frac{2C}{A + B},$$

where C is the number of tags (index terms) indexers I_1 and I_2 have in common and A and B is the size of their tag sets respectively.

In our experiments, before computing the number of terms in common, we stem each tag with the Porter (1980) stemmer. For example, the overlap C between the tag sets {*complex systems, network, small world*} and {*theoretical, small world, networks, dynamics*} consist of the two tags {*network, small world*}, and the consistency is 2×2/(3+4) = 0.57.

To compute the overall consistency of a particular indexer, this figure is averaged over all documents and co-indexers. There were no cases where the same user reassigned tags to the same articles, so computing intra-tagger consistency, although interesting, was not impossible.

To our knowledge, traditional indexing consistency metrics have not yet been applied to collaboratively tagged data. However, experiments on determining tagging quality do follow the same idea. For example, Xu *et al.* (2006) define an authority metric that assigns high scores to those users who match other users' choices on the same documents, in order to eliminate spammers.

2.3 Consistency of CiteULike taggers

In the collection of 180 documents tagged by 332 users described in Section 3.1, each tagger has 18 co-taggers on average, ranging from 2 to 129, and has indexed 1 to 25 documents. For each user we compute the consistency with all other users who tagged the same document. Consistency is then averaged across documents. We found that the distribution of per-user consistency resembles a power law with a few users achieving high consistency values and a long tail of inconsistent taggers. The maximum consis-

tency in this group is 92.3% and the average is 18.5%. The average consistency of the most prolific 70 indexers—those who have indexed at least five documents—is in the same range, namely 18.4%. The consistency of traditional approaches to free indexing is reported to be between 4% and 67%, with an average of 27% depending on what aids are used (Leininger, 2000).

It is instructive to consider the group of best taggers. We define these as the ones who (a) exhibit greater than average consistency with all others, and (b) are sufficiently prolific, i.e. have tagged at least five documents. There are 36 such taggers; Table 1 lists their consistency within this group. The average consistency they achieve as a group is 37.7%, which is the similar to the average consistency of professionals (Leininger, 2000).

The above consistency analysis provides insight into the tagging quality of the best CiteULike users, based on HighWire and Nature articles. For the purposes of this paper, it shows how the tagging community can be restricted to a best-performing group of taggers by measuring their consistency. This is helpful for testing the performance of automatic tagging (Section 4.4).

3 Automatic tagging with Maui

Maui is a general algorithm for automatic topical indexing based on the Kea system (Frank *et al.*, 1999).[1] It works in two stages: candidate selection and machine learning based filtering. In this paper, we apply it to automatic tagging. In the candidate selection stage, Maui first determines textual sequences defined by orthographic boundaries and splits these sequences into tokens. Then all n-grams up to a maximum length of 3 words that do not begin or end with a stopword are extracted as candidate tags. To reduce the number of candidates, all those that appear only once are discarded. This speeds up the training and the extraction process without impacting the results. In the filtering stage several features are computed for each candidate, which are then input to a machine learning model to obtain the probability that the candidate is indeed a tag.

Maui's architecture resembles that of many other supervised keyphrase extraction systems (Turney, 2000; Hulth 2004; Medelyan *et al.*, 2008). However, this architecture has not previously been applied to the task of automatic tagging.

3.1 Features indicating significance

We now describe the features used in the classification model to determine whether a phrase is likely to be a tag. We begin with three baseline features used in Kea (Frank *et al.*, 1999), and extend the set with three features that have been found useful in previous work. We also add three new features that have not been evaluated before: *spread*, *semantic relatedness* and inverse *Wikipedia linkage*. All Wikipedia-based features are computed using the WikipediaMiner toolkit.[2]

1. TF×IDF combines the frequency of a phrase in a particular document with its inverse occurrence frequency in general use (Salton and McGill, 1983). This score is high for rare phrases that appear frequently in a document and therefore are more likely to be significant.

2. Position of the first occurrence is computed as the relative distance of the first occurrence of the candidate tag from the beginning of the document. Candidates with very high or very low values are likely to be tags, because they appear either in the opening document parts such as title, abstract, table of contents, and introduction, or in the document's final sections such as conclusion and reference lists.

3. Keyphraseness quantifies how often a candidate phrase appears as a tag in the training corpus. Automatic tagging approaches utilize the same information: Mishne (2006) and Sood *et al.* (2006) automatically suggest tags previously assigned to similar documents. However, in Maui (as in Kea) this feature is just one component of the overall model. Thus if a candidate never appears as a keyphrase in the training corpus, it can still be extracted if its other feature values are significant enough.

4. Phrase length is measured in words. Generally speaking, the longer the phrase, the more specific it is. Training captures and quantifies the specificity preference in a given training corpus.

5. Node degree quantifies the semantic relatedness of a candidate tag to other candidates. Turney (2003) computes semantic relatedness using search engine statistics. Instead, following Medelyan *et al.* (2008), we utilize Wikipedia hyperlinks for this task. We first map each candidate phrase to its most common Wikipedia page. For example, the word *Jaguar* appears as a link anchor in Wikipedia 927 times. In 466 cases it links to the article *Jaguar cars*, thus the commonness of this mapping is 0.5. In 203 cases it links to the animal description, a commonness of

[1] Maui is open-source and available for download at *http://maui-indexer.googlecode.com*

[2] http://wikipedia-miner.sourceforge.net/

0.22. We compute the node degree of the corresponding Wikipedia article as the number of hyperlinks that connect it to other Wikipedia pages that have been identified for other candidate tags from the same document. A document that describes a particular topic will cover many related concepts, so high node degree—which indicates strong connectivity to other phrases in the same document—means that a candidate is more likely to be significant.

6. Wikipedia-based keyphraseness is the likelihood of a phrase being a link in the Wikipedia corpus. It divides the number of Wikipedia pages in which the phrase appears in the anchor text of a link by the total number of Wikipedia pages containing it. We multiply this number by the phrase's document frequency.

The new features proposed in this paper are the following:

7. Spread of a phrase is the distance between its first and last occurrences in a document. Both values are computed relative to the length of the document (see feature 2). High values help to determine phrases that are mentioned both in the beginning and at the end of a document.

8. Semantic relatedness of a phrase has already been captured as the node degree (see feature 5). However, recent research allows us to compute semantic relatedness with better techniques than mere hyperlink counts. Milne and Witten (2008) propose an efficient Wikipedia based approach that is nearly as accurate as human subjects at quantifying the relationship between two given concepts. Given a set of candidate phrases we determine the most likely Wikipedia articles that describe them (as explained in feature 5), and then determine the total relatedness of a given phrase to all other candidates. The higher the value, the more likely is the phrase to be a tag.

9. Inverse Wikipedia linkage is another feature that utilizes Wikipedia as a source of language usage statistics. Here, again given the most likely Wikipedia article for a given phrase, we count the number of other Wikipedia articles that link to it and normalize this value as in inverse document frequency:

$$IWL = -\log_2 \frac{linksTo(A_P)}{N}$$

where $linksTo(A_P)$ is the number of incoming links to the article A representing the candidate phrase P, and N is the total number of links in our Wikipedia snapshot (52M). This feature highlights those phrases that refer to concepts commonly used to describe other concepts.

3.2 Machine learning in Maui

In order to build the model, we use the subset of the CiteULike collection described in Section 3.1. For each document we know a set of tags that at least two users have agreed on. This is used as ground truth for building the model. For each training document, candidate phrases (i.e. n-grams) are identified and their feature values are calculated as described above.

Each candidate is then marked as a positive or negative example, depending on whether users have assigned it as a tag to the corresponding document. The machine-learning model is constructed automatically from these labeled training examples using the WEKA machine learning workbench. Kea (Frank *et al.*, 1999) uses the Naïve Bayes classifier, which implicitly assumes that the features are independent of each other given the classification. However, Kea uses only two or three features, whereas Maui combines nine features amongst which there are many obvious relationships, e.g. first occurrence and spread, or node degree and semantic relatedness. Consequently, we also consider bagged decision trees, which can model attribute interactions and do not require parameter tuning to yield good results. Bagging learns an ensemble of classifiers and uses them in combination, thereby often achieving significantly better results than the individual classifiers (Breiman, 1996). Different trees are generated by sampling from the original dataset with replacement. Like Naïve Bayes, bagged trees yield probability estimates that can be used to rank candidates.

To select tags from a new document, Maui determines candidate phrases and their feature values, and then applies the classifier built during training. This classifier determines the probability that a candidate is a tag based on relative frequencies observed from the training data.

4 Evaluation

Here we describe the data used in the experiments and the results obtained, addressing the following questions:

1. How does a state-of-the-art keyphrase extraction method perform on collaboratively tagged data, compared to a baseline automatic tagging method?
2. What is the performance of Maui with old and new features?
3. How consistent are Maui's tags compared to those assigned by human taggers?

		P	R	F
1	Top words based on TF×IDF	16.8	17.3	17.0
2	Top phrases based on TF×IDF	14.4	16.0	15.2
3	Kea (TF×IDF, 1st occur)	20.4	22.3	21.3
4	Kea (+keyphraseness)	**41.1**	**43.1**	**42.1**

Table 2. Baseline auto-tagging approach vs. Kea

4.1 Evaluation method

The evaluation was performed using a set of 180 documents, described in Section 3.1, each tagged with at least three tags on which two users have agreed. In the following, unless explicitly stated otherwise, these are the only tags we use. We consider them to be ground truth. There are on average five such tags per document, and our goal is to extract tag sets that contain them all.

We regard a predicted tag as "correct" if it matches one of the ground truth tags after using the Porter stemmer. We measure performance by computing Precision (the percentage of correct extracted tags out of all extracted), Recall (the percentage of correct extracted tags out of all correct) and F-Measure (the harmonic mean of the two). Given the set {*yeast* (4), *network* (3), *regulation* (2), *metabolic* (2)} of ground truth tags, where the numbers in parenthesis show how many users have assigned each one, and the set {*network*, *metabolic*, *regulatory*, *ChIP-chip*, *transcription*} of predicted tags, three out of five predicted terms are correct, yielding a precision of 60%, and three out of four ground-truth terms are extracted, a recall of 75%. The F-measure combining the two values is 67%.

The reported precision and recall values are averaged over all test documents. We use 10-fold cross-validation for evaluation, which allows us to use all 180 documents as test documents without introducing optimistic bias in the performance measures obtained.

The results obtained in Sections 4.2 and 4.3 using this evaluation provide answers to the first two questions above. To answer the third we compare the indexing consistency of Maui to that of CiteULike users in Section 4.4. Here, we consider the assigned tag sets individually and compute the consistency of Maui with each tagger as described in Section 3.2. We compare Maui both to all 332 users who tagged these documents, and to the 36 best taggers identified in Section 3.3.

4.2 Keyphrase extraction vs. auto-tagging

As noted earlier, Brooks and Montanez (2006) automatically determine tags by extracting terms with the highest TF×IDF values for each post and argue that their quality is perhaps better than

		P	R	F
1	TFxIDF	14.4	16	15.2
2	1st occurrence	5.4	5.4	5.4
3	Keyphraseness	25.2	26.3	25.5
4	Length	2.1	2.1	2.1
5	Node degree	8.3	9.0	8.6
6	Wikipedia keyphraseness	16.9	18.3	17.6
7	Spread	12.1	13.0	12.5
8	Semantic relatedness	7.1	7.3	7.2
9	Inverse Wikipedia linkage	7.3	6.8	7.0

Table 3. Evaluation of individual features

that of manual tags. Note that they only use one-word tags. We evaluate this approach using our 180 test documents and cross-validation, and compare the top five extracted tags with those assigned manually. Comparing the first two rows of Table 2 shows that using multi-word phrases as candidate tags (Section 4) is less accurate than using single words, which gives an overall F-Measure of 17%. Multi-words have higher TF×IDF values, but single words are the majority among the users' tags. The length feature applied in the next section helps to capture this characteristic, without compromising Maui's ability to assign correct multi-words tags.

Adding a second feature, the position of the first occurrence, and using Kea's Naïve Bayes model to learn their conditional distribution, improves the results by 5 percentage points (row 3). Adding the keyphraseness feature (row 4) nearly doubles the F-Measure, from 21.3 to 42.1%. This shows that CiteULike users tend to re-assign existing tags.

4.3 Maui with additional features

To evaluate Maui let us first consider the individual performance of old and new features, as shown in Table 3. Rows 1 to 3 evaluate the standard features used by Frank *et al.* (1999); Rows 4 to 6 evaluate features that were previously used in Kea for controlled indexing (Medelyan *et al.*, 2006) and which we have adapted in Maui for free indexing. Rows 7 to 9 evaluate the three new features of Maui. The values can be compared to keyphrase extraction by chance (F-Measure = 1%) and to the multi-word TF×IDF baseline in Table 2, row 2 (F-Measure = 15.2%). The strength of these features varies from 2.1 to 25.5% (F-Measure). The strongest ones are keyphraseness, Wikipedia keyphraseness, TF×IDF and spread.

Table 4 demonstrates Maui's performance when the features are combined and shows how the two different classifiers, Naïve Bayes (left) and bagged decision trees (right), compare to

		Naïve Bayes			Bagged decision trees		
		P	R	F	P	R	F
1	Features 1 – 3	**41.1**	**43.1**	**42.1**	40.3	42.2	41.2
2	Features 1 – 6	38.9	41.1	40.0	40.3	42.6	41.4
3	Features 1 – 3, 7 – 9	39.3	41.1	40.2	43.7	46.2	44.9
4	Features 1 – 9	37.6	39.6	38.6	**45.7**	**48.6**	**47.1**

Table 4. Combining all features in Maui

each other. The baseline in row 1 (left) shows Kea's performance, using TF×IDF, first occurrence, keyphraseness and Naïve Bayes to combine them (same as row 4 in Table 2). Using decision trees with these three features does not improve the performance (row 1, right). The following row combines the three original features with length, node degree and Wikipedia-based keyphraseness. In contrast to previous research (Medelyan *et al.*, 2008), in this setting we do not observe an improvement with either Naïve Bayes or bagged decision trees. In row 3 we combine the three original features with the three new ones introduced in this work. While Naïve Bayes' values are lower than the baseline, with bagged decision trees Maui's F-Measure improves from 41.2 to 44.9%. The best results are obtained by combining all nine features, again using bagged decision trees, giving in row 4 (right) a notably improved F-Measure of 47.1%. The recall of 48.6% shows that we match nearly half of all tags on which at least two human taggers have agreed.

Given this best combination of features, we eliminate each feature one by one starting from the individually weakest feature, in order to determine the contribution of each feature to this overall result. Table 5 compares the values and only bagged decision trees are used this time. The 'Difference' column quantifies the difference between the best F-Measure achieved with all 9 features and excluding the one that is examined in that row. Interestingly, one of the strongest features, TF×IDF, is the one that contributes the least when all features are combined, while the contribution of the strongest feature—keyphraseness—is, as expected, the highest, adding 16.9 points. The second most important feature is Wikipedia keyphraseness, contributing 4 percentage points to the overall result.

Features	F-Measure	Difference
All 9 Features	47.1	
– Length	45	2.1
– 1st occurrence	45.6	1.5
– Inverse Wikip linkage	45.1	2
– Semantic relatedness	45.4	1.7
– Node degree	46	1.1
– Spread	46.4	0.7
– TFxIDF	46.8	0.3
– Wikip keyphraseness	43.1	4
– Keyphraseness	30.2	16.9
Non-Wikip features	41.7	5.4

Table 5. Evaluation using feature elimination

Since some of the features in the best performing combination rely on Wikipedia as a knowledge source, it is interesting to determine Wikipedia's exact contribution. The last row of Table 5 combines the following features: TF×IDF, first occurrence, keyphraseness, length and spread. The F-Measure is 5.4 points lower than that of Maui with all 9 features combined. Therefore, the contribution of Wikipedia-based features is significant.

4.4 Maui's consistency with human taggers

In Section 2.3 we discussed the indexing consistency of CiteULike users on our data. There are a total of 332 taggers and their consistency with each other is 18.5%. Now, we use results obtained with Maui during the cross-validation, when all 9 features and bagged decision trees are used (Table 4, row 4, right; see examples in Table 5), and compute how consistent Maui is with each human user, based on whatever document this user has tagged. Then we average the results to obtain the overall consistency with all 332 users.

Maui's consistency with the 332 human taggers ranges from 0 to 80%, with an average of 23.8%. The only cases where very low consistency was achieved are those where the human has only assigned a few tags per document (one to three), or has some idiosyncratic tagging behavior (for example, one tagger adds the word *key* in front of most tags). Still, with an average of 23.8%, Maui's performance is over 5 points higher than that of an average CiteULike tagger (18.5%)—and note this group only includes taggers who have at least two co-taggers.

In Section 2.3 we were also able to determine a smaller group of users who perform best and are most prolific. This group consists of 36 taggers whose consistency exceeds the average of the original 332 users. These 36 taggers have tagged a total of 143 documents with an average consistency of 37.6%. Maui's consistency with

Document	86865. Neural correlates of decision variables in parietal cortex. Platt and Glimcher. *Nature* 400,15 (1999)	44. Exploring complex networks. Strogatz. *Nature* 410, 8 (2001)	353537. Computational roles for dopamine in behavioural control. Montague et al. *Nature* 431, 14 (2004)	101. Network motifs: simple building blocks of complex networks. Milo et al. *Science* 298, 824 (2002)
Tags assigned by CiteULike taggers	**decision making** **decisionmaking** **lip** **monkey** **neurophysiology** **reward** ***Idiosyncratic:*** brain, choice, cortex, decision, electrophysiology, eye-movements, limitations, monkeys, neuroeconomics, neurons, neuroscience, other, ppc, quals, reinforcementlearning	**complex** **complexity** **complex networks** **graph** **networks** **review** **small world** **social networks** **survey** ***Idiosyncratic:*** 2001, adaptive systems, bistability, coupled oscillator, graph mining, graphs, explorig, network biological, neurons, strogatz	**dopamine** **neuroscience** **reinforcement learning** **review** ***Idiosyncratic:*** action selection, attention, behavior, behavioral control, cognitive control, learning, network, reinforcementlearning, reward, td model	**applied math** **combinatorics** **complexity** **motifs** **network** **original** **sub graph pattern** ***Idiosyncratic:*** 2002, datamining, data mining, graphs, link analysis, modularity, net paper, patterns, protein, science, sysbio, web characterization, web graph
Tags assigned by Maui	**cortex** **decision** **lip** **monkey** visual	**complex networks** **networks** **review** synchronization **graph**	**dopamine** **learning** **neuroscience** **review** **reward**	complex networks **network** **motifs** gene **complex**

Table 6. Tags assigned by CiteULike taggers and Maui to four sample documents

these taggers ranges from 11.5% to 56%, with an average of 35%. This places it only 2.6 percentage points behind the average performance of the best CiteULike taggers. In fact, it outperforms 17 of them (cf. Table 1).

4.5 Examples

Table 6 compares Maui with some of CiteULike's best human taggers on four randomly chosen test documents. Boldface in the taggers' row indicates a tag that has been chosen by at least two other human taggers; the remaining tags have been chosen by just one human. Boldface in Maui's row shows tags that match human tags. For each document Maui extracts several tags assigned by at least two humans. The other tags it chooses are generally chosen by at least one human tagger, and even if not, they are still related to the main theme of the document.

5 Discussion and related work

It is possible to indirectly compare the results of several previously published automatic tagging approaches with Maui's. For each paper, we compute Maui's results in settings closest to the reported ones.

Brooks and Montanez (2006) extract terms with the highest TF×IDF values as tags for posts on *technorati.com*. They do not report precision and recall values for their system, but our reimplementation resulted in precision of 16.8% and recall of 17.3% for the top five assigned tags, compared to those agreed to by at least two CiteULike users on 180 documents. Adding eight additional features and combining them using machine learning gives a clear improvement—Maui achieves 45.7% and 48.7% precision and recall respectively.

Mishne (2006) uses TF×IDF-weighted terms as full-text queries to retrieve posts similar to the one being analyzed. Tags assigned to these posts are analyzed to retrieve the best ones using clustering and heuristic ranking; tags assigned by the given user receive extra weight. Mishne performs manual evaluation on 30 short articles and reports precision and recall for the top ten tags of 38% and 47% respectively. We matched Maui's top ten terms to all tags assigned to 180 documents automatically and obtained precision and recall of 44% and 29% respectively. (We believe that manual rather than automatic evaluation would be likely to give a far more favorable assessment of our system.)

Chirita *et al.* (2007) aim to extract personalized tags. Given a web page, they first retrieve

similar documents stored on the user's desktop and then determine keywords for these documents. They evaluate different term scoring techniques, such as term and document frequency, lexical dispersion, sentence scoring, and term co-occurrence. Like the Kea algorithm, the best formula combines term frequency with the position of the first occurrence of the term, normalized by page length. It yields a precision of 80% for the top four tags assigned to 30 large websites (32Kbytes), again evaluated manually. Our documents are considerably longer (47Kbytes) and thus more difficult to work with, nevertheless Maui achieves only slightly lower values, from 66% to 80%, when evaluating automatically against user-assigned tags. (The above caveat regarding automatic and manual assessment applies here too.)

Budura *et al.* (2008) develop a scoring formula that combines three features (tag frequency, tag co-occurrence and document similarity) and manually evaluate it on ten CiteULike documents. Their precision for the top three to five tags ranges from 66% to 77%, slightly worse than in our paper (66% to 80%).

The only reported automatic evaluation of tags was found in Sood *et al.* (2006), where TagAssist was tested on 1000 blog posts. This algorithm is similar to Mishne's (2006), but uses centroid-based clustering. Exact matching of TagAssist's tags against existing ones yielded precision and recall of 13.1% and 22.8% respectively. This is substantially lower than Maui's 45.75% and 48.7% obtained with best settings (Section 4.3).

Note that this indirect comparison does not reveal the true ranking of approaches, because their task definitions and test sets are slightly different. It would be interesting to compare other systems on the multiple tagger set described in this paper, as we believe this would more objectively reflect the performance of humans and algorithms.

6 Conclusions

This paper has introduced a systematic way of evaluating automatic tagging techniques without the need for manual inspection. We have shown how documents with multiple tag sets can be used in conjunction with a standard consistency measure to identify a robust test corpus for these techniques. Based on the evaluation methodology developed, we have shown that machine-learning-based automatic keyphrase extraction produces tag sets that exhibit consistency on a par with that achieved by the best human taggers. Our results also show a substantial improvement on an existing automatic tagging approach based on TF×IDF, and the results compare well to other systems.

The success of automatic keyphrase extraction depends primarily on the quality of the features that are provided to the machine learning algorithm involved. In this paper we have evaluated nine different features, including two novel Wikipedia-based semantic features, and found that their combination used in conjunction with bagged decision trees produces the best performance.

References

Breiman, L. 1996. Bagging predictors. *Machine Learning* 24(2): 123–140.

Brooks, C. H. and N. Montanez. 2006. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *Proc. Int. Conf. on World Wide Web,* Edinburgh, UK. pp. 625–632. New York, NY, USA. ACM Press.

Budura, A., S. Michel, P. Cudre-Mauroux, and K. Aberer. 2008. To tag or not to tag - harvesting adjacent metadata in large-scale tagging systems. In *Proc. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Singapore. pp. 733–734. New York, NY, USA: ACM Press.

Chirita, P. A., S. Costache, W. Nejdl, and S. Handschuh. 2007. P-tag: large scale automatic generation of personalized annotation tags for the web. In *Proc. Int. Conf. on World Wide Web,* Banff, Canada. pp. 845–854. New York, NY, USA: ACM Press.

Frank, E., G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proc. of the 16th International Joint Conference on Artificial Intelligence*, Stockholm, Sweden. pp. 668–673. San Francisco, CA: Morgan Kaufmann Publishers.

Golder, S. A. and B. A. Huberman. 2006. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2): 198–208.

Halpin, H., V. Robu, and H. Shepherd. 2007. The complex dynamics of collaborative tagging. In *Proc. Int. Conf. on World Wide Web,* pp. 211–220. New York, NY, USA: ACM Press.

Heymann, P., D. Ramage, and H. Garcia-Molina. 2008. Social tag prediction. In *Proc. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Singapore. pp. 531–538. New York, NY, USA: ACM Press.

Hulth, A. 2004. *Combining machine learning and natural language processing for automatic keyword extraction.* Ph.D. thesis, Dep. of Computer and Systems Sciences, Stockholm University.

Leonard, L. E. 1975. *Inter-indexer consistency and retrieval effectiveness: measurement of relationships*. Ph.D. thesis, Grad. School of Library Science, Univ. of Illinois, Urbana-Champaign, IL.

Leininger, K. 2000. Interindexer consistency in PsycInfo. *Journal of Librarianship and Information Science* 32(1): 4–8.

Medelyan, O., I. H. Witten and D. Milne. 2008. Topic indexing with Wikipedia. In *Proc. of AAAI'08 Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy*, Chicago, USA. pp. 19–24.

Mishne, G. 2006. Autotag: a collaborative approach to automated tag assignment for weblog posts. In *Proc. Int. Conf. on World Wide Web*, Edinburgh, UK. pp. 953–954. New York, NY, USA. ACM Press

Porter, M. F. 1980. An algorithm for suffix stripping, *Program*, 14(3): 130−137.

Rolling, L. 1981. Indexing Consistency, Quality and Efficiency. *Information Processing & Management* 17(2): 69–76.

Salton, G. and M. J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill New York.

Sigurbjörnsson, B. and R. van Zwol. 2008. Flickr tag recommendation based on collective knowledge. In *Proc. Int. Conf. on World Wide Web,* Beijing, China. pp. 327–336. New York, NY, USA: ACM Press.

Sood, S., K. Hammond, S. Owsley, and L. Birnbaum. 2007. TagAssist: Automatic tag suggestion for blog posts. of *Int. Conf. on Weblogs and Social Media*, Boulder, Colorado. Menlo Park, CA.

Turney, P. D. 2003. Coherent keyphrase extraction via web mining. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence,* Acapulco, Mexico. pp. 434–439. San Francisco, CA: Morgan Kaufmann Publishers.

Xu, Z., Fu, Y., Mao, J., and D. Su. 2006. Towards the Semantic Web: Collaborative tag suggestions. In *Proc. Collaborative Web Tagging Workshop at the Int. Joint Conf. on Artificial Intelligence,* Stockholm, Sweden.

Supervised Learning of a Probabilistic Lexicon of Verb Semantic Classes

Yusuke Miyao
University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
yusuke@is.s.u-tokyo.ac.jp

Jun'ichi Tsujii
University of Tokyo
University of Manchester
National Center for Text Mining
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
tsujii@is.s.u-tokyo.ac.jp

Abstract

The work presented in this paper explores a supervised method for learning a probabilistic model of a lexicon of VerbNet classes. We intend for the probabilistic model to provide a probability distribution of verb-class associations, over known and unknown verbs, including polysemous words. In our approach, training instances are obtained from an existing lexicon and/or from an annotated corpus, while the features, which represent syntactic frames, semantic similarity, and selectional preferences, are extracted from unannotated corpora. Our model is evaluated in type-level verb classification tasks: we measure the prediction accuracy of VerbNet classes for unknown verbs, and also measure the dissimilarity between the learned and observed probability distributions. We empirically compare several settings for model learning, while we vary the use of features, source corpora for feature extraction, and disambiguated corpora. In the task of verb classification into all VerbNet classes, our best model achieved a 10.69% error reduction in the classification accuracy, over the previously proposed model.

1 Introduction

Lexicons are invaluable resources for semantic processing. In many cases, lexicons are necessary to restrict a set of semantic classes to be assigned to a word. In fact, a considerable number of works on semantic processing implicitly or explicitly presupposes the availability of a lexicon, such as in word sense disambiguation (WSD) (McCarthy et al., 2004), and in token-level verb class disambiguation (Lapata and Brew, 2004; Girju et al., 2005; Li and Brew, 2007; Abend et al., 2008). In other words, those methods are heavily dependent on the availability of a semantic lexicon. Therefore, recent research efforts have invested in developing semantic resources, such as WordNet (Fellbaum, 1998), FrameNet (Baker et al., 1998), and VerbNet (Kipper et al., 2000; Kipper-Schuler, 2005), which greatly advanced research in semantic processing. However, the construction of such resources is expensive, and it is unrealistic to presuppose the availability of full-coverage lexicons; this is the case because unknown words always appear in real texts, and word-semantics associations may vary (Abend et al., 2008).

This paper explores a method for the supervised learning of a probabilistic model for the VerbNet lexicon. We target the automatic classification of arbitrary verbs, including polysemous verbs, into all VerbNet classes; further, we target the estimation of a probabilistic model, which represents the saliences of verb-class associations for polysemous verbs. In our approach, an existing lexicon and/or an annotated corpus are used as the training data. Since VerbNet classes are designed to represent the distinctions in the syntactic frames that verbs can take, features, representing the statistics of syntactic frames, are extracted from the unannotated corpora. Additionally, as the classes represent semantic commonalities, semantically inspired features, like distributionally similar words, are used. These features can be considered as a generalized representation of verbs, and we expect that the obtained probabilistic model predicts VerbNet classes of the unknown words.

Our model is evaluated in two tasks of type-level verb classification: one is the classification of monosemous verbs into a small subset of the classes, which was studied in some previous works (Joanis and Stevenson, 2003; Joanis et al., 2008). The other task is the classification of all verbs into the full set of VerbNet classes, which has not yet

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1328–1337,
Singapore, 6-7 August 2009.

been attempted. In the experiments, training instances are obtained from VerbNet and/or SemLink (Loper et al., 2007), while features are extracted from the British National Corpus or from Wall Street Journal. We empirically compare several settings for model learning by varying the set of features, the source domain and the size of a corpus for feature extraction, and the use of the token-level statistics obtained from a manually disambiguated corpus. We also provide the analysis of the remaining errors, which will lead us to further improve the supervised learning of a probabilistic semantic lexicon.

Supervised methods for automatic verb classification have been extensively investigated (Stevenson et al., 1999; Stevenson and Merlo, 1999; Merlo and Stevenson, 2001; Stevenson and Joanis, 2003; Joanis and Stevenson, 2003; Joanis et al., 2008). However, their focus has been limited to a small subset of verb classes, and a limited number of monosemous verbs. The main contributions of the present work are: i) to provide empirical results for the automatic classification of all verbs, including polysemous ones, into all VerbNet classes, and ii) to empirically explore the effective settings for the supervised learning of a probabilistic lexicon of verb semantic classes.

2 Background

2.1 Verb lexicon

Levin's (1993) work on verb classification has broadened the field of computational research that concerns the relationships between the syntactic and semantic structures of verbs. The principal idea behind the work is that the meanings of verbs can be identified by observing possible syntactic frames that the verbs can take. In other words, with the knowledge of syntactic frames, verbs can be semantically classified. This idea provided the computational linguistics community with criteria for the definition and the classification of verb semantics; it has subsequently resulted in the research of the induction of verb classes (Korhonen and Briscoe, 2004), and the construction of a verb lexicon based on Levin's criteria.

VerbNet (Kipper et al., 2000; Kipper-Schuler, 2005) is a lexicon of verbs organized into classes that share the same syntactic behaviors and semantics. The design of classes originates from Levin (1993), though the design has been considerably reorganized and extends beyond the original classification. The classes therefore cover more English verbs, and the classification should be more consistent (Korhonen and Briscoe, 2004; Kipper et al., 2006).

43 Emission
- **43.1 Light Emission**
  beam, glow, sparkle, ...
- **43.2 Sound Emission**
  blare, chime, jangle, ...

...

44 *Destroy*
annihilate, destroy, ravage, ...

45 Change of State

...

47 Existence
- **47.1 *Exist***
  exist, persist, remain, ...
- **47.2 Entity-Specific Modes Being**
  bloom, breathe, foam, ...
- **47.3 Modes of Being with Motion**
  jiggle, sway, waft, ...

...

Figure 1: VerbNet classes

43.2 Sound Emission
```
Theme V
Theme V P:loc Location
P:loc Location V Theme
there V Theme P:loc Location
Agent V Theme
Theme V Oblique
Location V with Theme
```

47.3 Modes of Being with Motion
```
Theme V
Theme V P:loc Location
P:loc Location V Theme
there V Theme
Agent V Theme
```

Figure 2: Syntactic frames for VerbNet classes

The current version of VerbNet includes 270 classes.[1] Figure 1 shows a part of the classes of VerbNet. The top-level categories, e.g. **Emission** and ***Destroy***, represent a coarse classification of verb semantics. They are further classified into verb classes, each of which expresses a group of verbs sharing syntactic frames. Figure 2 shows an excerpt from VerbNet, which represents the possible syntactic frames for the **Sound Emission** class, including "*chime*" and "*jangle*," and the **Modes of Being with Motion** class, including "*jiggle*" and "*waft*." In this figure, each line represents a syntactic frame, where `Agent`,

[1] Throughout this paper, we refer to VerbNet 2.3. Subclasses are ignored in this work, following the setting of Abend et al. (2008).

...the walls still shook;VN=47.3 and an evacuation alarm blared;VN=43.2 outside.

Suddenly the woman begins;VN=55.1 swaying ;VN=47.3 and then ...

Figure 3: An excerpt from SemLink

Theme, and Location indicate the thematic roles, V denotes a verb, and P specifies a preposition. P:loc defines locative prepositions such as: "*in*" and "*at*." For example, the second syntactic frame of **Sound Emission**, i.e., Theme V P:loc Location, corresponds to the following sentence:

1. The coins *jangled* in my pocket.

Theme corresponds to "*the coins*," V to "*jangled*," P:loc to "*in*," and Location to "*my pocket*."

While VerbNet provides associations between verbs and semantic classes, SemLink (Loper et al., 2007) additionally provides mappings among VerbNet, FrameNet (Baker et al., 1998), PropBank (Palmer et al., 2005), and WordNet (Fellbaum, 1998). Since FrameNet and PropBank include annotated instances of sentences, SemLink can be considered as a corpus annotated with VerbNet classes. Figure 3 presents some annotated sentences obtained from SemLink. For example, the annotation "blared;VN=43.2" indicates that the occurrence of "blare" in this context is classified as **Sound Emission**.

2.2 Related work

There has been much research effort invested in the automatic classification of verbs into lexical semantic classes, in a supervised or unsupervised way. The present work inherits the spirit of the supervised approaches to verb classification (Stevenson et al., 1999; Stevenson and Merlo, 1999; Merlo and Stevenson, 2001; Stevenson and Joanis, 2003; Joanis and Stevenson, 2003; Joanis et al., 2008). Our learning framework basically follows the above listed works: features are obtained from an unannotated (automatically parsed) corpus, and gold verb-class associations are used as training instances for machine learning classifiers, such as decision trees and support vector machines. However, those works targeted a small subset of Levin classes, and a limited number of monosemous verbs; for example, Merlo and Stevenson (2001) studied three classes and 59 verbs, and Joanis et al. (2008) focused on 14 classes and 835 verbs. Although these works provided a theoretical framework for supervised verb classification, their results were not readily available for practical applications, because of the limitation in the coverage of the targeted classes/verbs on real texts. On the contrary, we target the classification of arbitrary verbs, including polysemous verbs, into all VerbNet classes (270 in total). In this realistic situation, we will empirically compare settings for model learning, in order to explore effective conditions to obtain better models.

Another difference from the aforementioned works is that we aim at obtaining a probabilistic model, which represents *saliences* of classes of polysemous verbs. Lapata and Brew (2004) and Li and Brew (2007) focused on this issue, and described methods for inducing probabilities of verb-class associations. The obtained probabilistic model was intended to be incorporated into a token-level disambiguation model. Their methods claimed to be unsupervised, meaning that the induction of a probabilistic lexicon did not require any hand-annotated corpora. In fact, however, their methods relied on the existence of a full-coverage lexicon, both in training and running time. In their methods, a lexicon was necessary for restricting possible classes to which each word belongs. Since most verbs are associated with only a couple of classes, such a restriction significantly reduces the search space, and the problem becomes much easier to solve. This presupposition is implicitly or explicitly used in other semantic disambiguation tasks (McCarthy et al., 2004), but it is unrealistic for practical applications.

Clustering methods have also been extensively researched for verb classification (Stevenson and Merlo, 1999; Schulte im Walde, 2000; McCarthy, 2001; Korhonen, 2002; Korhonen et al., 2003; Schulte im Walde, 2003). The extensive research is in large part due to the intuition that the set of classes could not be fixed beforehand. In particular, it is often problematic to define a static set of semantic classes. However, it is reasonable to assume that the set of VerbNet classes is fixed, because Levin-type classes are more static than ontological classes, like in WordNet synsets. Therefore, we can apply supervised classification methods to our task. It is true that the current VerbNet classes are imperfect and require revisions, but in this work we adopt them as they are, because as

time advances, more stable classifications will become available.

The problem focused in this work has a close relationship with automatic thesaurus/ontology expansion. In fact, we evaluate our method in the task of automatic verb classification, which can be considered as lexicon expansion. The most prominent difference of the present work from thesaurus/ontology expansion is that the number of classes is much smaller in our problem, and the set of verb classes can be assumed to be fixed. These characteristics indicate that our problem is easier and more well-defined than is the case for automatic thesaurus/ontology expansion.

Supervised approaches to token-level verb class disambiguation have recently been addressed (Girju et al., 2005; Abend et al., 2008), largely owing to SemLink. Their approaches fundamentally follow traditional supervised WSD methods: extracting features representing the context in which the target word appears, and training a classification model with an annotated corpus. While those works achieved an impressive accuracy (more than 95%), the results may not necessarily indicate the method's effectiveness; rather, it may imply the importance of a lexicon. In fact, these works restrict their target to verb tokens, in which the correct class exists in a given lexicon, and they only consider candidate classes that are registered in the lexicon. This setting reduces the ambiguity significantly, and the problem becomes much easier to handle; for example, approximately half of verb tokens are monosemous in their setting. Thus, a simple baseline achieves very high accuracy figures. However, in our preliminary experiment on token-level verb classification with unknown verbs, we found that the accuracy for unknown verbs (i.e., lemmas not included in the VerbNet lexicon) is catastrophically low. This indicates that VerbNet and SemLink are insufficient for unknown verbs, and that we cannot expect the availability of a full-coverage lexicon in the real world. Instead of a static lexicon, our probabilistic model is intended to be used as a prior distribution for the token-level disambiguation, as in Lapata and Brew (2004)'s model.

3 A probabilistic model for verb semantic classes

In this work, supervised learning is applied to the probabilistic modeling of a lexicon of verb semantic classes. We do not presuppose the existence of a full-coverage lexicon; instead, we use an existing lexicon for the training data. Combined with features extracted from unannotated corpora, a probabilistic model is learned from the existing lexicon. Like other supervised learning applications, our probabilistic lexicon can predict classes for words that are not included in the original lexicon.

Our model is defined in the following way. We assume that the set, C, of verb classes is fixed, while a set of verbs is unfixed. With this assumption, probabilistic modeling can be reduced to a classification problem. Specifically, the goal is to obtain a probability distribution, $p(c|v)$, of verb class $c \in C$ for a given verb (lemma) v. We can therefore apply well-known supervised learning methods to estimate $p(c|v)$.

This probability is modeled in the form of a log-linear model.

$$p(c|v) = \frac{1}{Z} \exp\left(\sum_i \lambda_i f_i(c, v)\right),$$

where $f_i(c, v)$ are features that represent characteristics of c and v, and λ_i are model parameters that express weights of the corresponding features.

Model parameters can be estimated when *training instances*, i.e., pairs $\langle c, v\rangle$, and *features*, $f_i(c, v)$, for each instance are given. Therefore, what we have to do is to prepare the training instances $\langle c, v\rangle$, and effective features $f_i(c, v)$ that contribute to the better estimation of probabilities. In token tagging tasks, both training instances and features are extracted from annotated corpora. However, since our goal is the probabilistic modeling of a lexicon, we have to determine how to derive the training instances and features for lexicon entries, to be discussed in the next section.

For the parameter estimation of log-linear models, we applied the stochastic gradient descent method. A hyperparameter for l_2-regularization was tuned to minimize the KL-divergence (see Section 4.4) for the development set.

4 Experiment design

In this work, we empirically compare several settings for the learning of the above probabilistic model, in the two tasks of automatic verb classification. In what follows, we explain the training/test data, corpora for extracting features, and the design of the features and evaluation tasks. The measures for evaluation are also introduced.

1	sound_emission-43.2	chime
0.5	sound_emission-43.2	blare
0.5	manner_speaking-37.3	blare
0.5	modes_of_being_with_motion-47.3	sway
0.5	urge-58.1	sway

1	sound_emission-43.2	chime
0.7	sound_emission-43.2	blare
0.3	manner_speaking-37.3	blare
0.6	modes_of_being_with_motion-47.3	sway
0.4	urge-58.1	sway

Figure 4: Training instances obtained from VerbNet (upper) and VerbNet+SemLink (lower)

4.1 Data

As our goal is the supervised learning of a lexicon of verb semantic classes, VerbNet is used as the training/test data. In addition, since we aim at representing the saliences of verb-class associations with probabilities, the gold probabilities are necessary. For this purpose, we count the occurrences of each verb-class association in the VerbNet-PropBank token mappings in the subset of the SemLink corresponding to sections 2 through 21 of Penn Treebank (Marcus et al., 1994). Frequency counts are normalized for each lemma, with the Laplace smoothing (the parameter is 0.5).

In this work, we compare the two settings for creating training instances. By comparing the results of these settings, we evaluate the necessity of an annotated corpus for learning a probabilistic lexicon of verb semantic classes.

VerbNet We collect all $\langle c, v \rangle$ pairs registered in VerbNet. For each v, all of the associated classes are assigned equal weights (see the upper part of Figure 4).

VerbNet+SemLink Each pair $\langle c, v \rangle$ in VerbNet is weighted by the normalized frequency obtained from SemLink (see the lower part of Figure 4).

Because VerbNet classes represent groups of syntactic frames, and it is impossible to guess the verb class by referring to only one occurrence in a text, it is necessary to have statistics over a sufficient amount of a corpus. Hence, features are extracted from a large unannotated corpus. In this paper, we use the following two corpora:

WSJ Wall Street Journal newspaper articles (around 40 million words).

BNC British National Corpus, which is a balanced corpus of around 100 million words.

In addition to the variance of the corpus domains, we vary the size of the corpus to observe the effect of increasing the corpus size. These corpora are automatically parsed by Enju 2.3.1 (Miyao and Tsujii, 2008), and the features are extracted from the parsing results.

4.2 Features

Levin-like classes, including VerbNet, are designed to represent distinctions in syntactic frames and alternations. Hence, if we were given the perfect knowledge of the possible syntactic frames, verbs can be classified into the correct classes almost perfectly (Dorr and Jones, 1996). Previous works thus proposed features that express the corpus statistics of syntactic frames. However, class boundaries are subtle in some cases; several classes share syntactic frames with each other to a large extent.

For example, the classes shown in Figure 2 have very similar syntactic frames. The difference is indicated in the last two frames of **Sound Emission**, although they appear much less frequently in real texts. Therefore, it is difficult to accurately capture the distinctions between these classes, if we are only provided with the statistics of the syntactic frames that appear in real texts. In this case, however, it is easy to observe that the verbs of these classes have different selectional preferences; that is, the `Theme` of **Sound Emission** verbs would be objects that make sounds, while the `Theme` of **Modes of Being with Motion** is likely to be objects that move.[2] Although Levin's classification initially focused on syntactic alternations, the resulting classes represent some semantic commonalities. Hence, it would be reasonable to design features that capture such semantic characteristics.

In this work, we re-implemented the following features proposed by Joanis et al. (2008) as the starting point.

Syntactic slot Features to count the occurrences of each syntactic slot, such as subject, object, and prepositional phrases. For the subject slot, we also count its transitive and intransitive usages separately. Additionally, we count the appearances of reflexive pronouns and semantically empty constituents (*it* and

[2] Syntactic frames in VerbNet include specifications of selectional preferences, such as *animate* and *place*, although we do not explicitly use them, because it is not apparent to determine the members of these semantic classes.

Syntactic slot	`subj:0.885` `intrans-subj:0.578`
Slot overlap	`overlap-subj-obj:0.299` `overlap-obj-in:0.074`
Tense, voice, aspect	`pos-VBG:0.307` `pos-VBD:0.290`
Animacy	`anim-subj:0.244` `anim-obj:0.057`
Slot POS	`subj-PRP:0.270` `subj-NN:0.270`
Syntactic frame	`NP_V:0.326` `NP_V_NP:0.307`
Similar word	`sim-rock:0.090` `sim-swing:0.083`
Slot class	`subj-C82:0.219` `obj-C12:0.081`

Figure 5: Example of features for "sway"

there). Differently from Joanis et al. (2008), we consider non-nominal arguments, such as sentential and adjectival complements.

Slot overlap Features to measure the overlap in words (lemmas) between two syntactic slots of the verb. They are intended to approximate argument alternations, such as the ergative alternation. For example, for the alternation "*The sky cleared*"/"*The clouds cleared from the sky*," a feature to indicate the overlap between the subject slot and the *from* slot is added (Joanis et al., 2008). The value of this feature is computed by the method of Merlo and Stevenson (2001).

Tense, voice, aspect Features to approximate the tendency of the tense, voice, and aspect of the target verb. The Penn Treebank POS tags for verbs (VB, VBP, VBZ, VBG, VBD, and VBN) are counted. In addition, included are the frequency of the co-occurrences with an adverb or an auxiliary verb, and the count of usages as a noun or an adjective.

Animacy Features to measure the frequency of animate arguments for each syntactic slot. Personal pronouns except *it* are counted as animate, following Joanis et al. (2008), while named entity recognition was not used.

Examples of these features are shown in Figure 5. For details, refer to Joanis et al. (2008).

The above features mainly represent syntactic behaviors of target verbs. Since our target classes are broader than in the previous works, we further enhance the syntactic features. Additionally, as discussed above, semantically motivated features may present strong clues to distinguish among syntactically similar classes. We therefore include the following four types of feature; the first two are syntactic, while the other two are intended to capture semantic characteristics:

Slot POS In addition to the syntactic slot features, we add features that represent a combination of a syntactic slot and the POS of its head word. Since VerbNet includes extended classes that take verbal and adjectival arguments, the POSs of arguments would provide a strong clue to discriminate among these syntactic frames.

Syntactic frame The number of arguments and their syntactic categories. This feature was mentioned as a baseline in Joanis et al. (2008), but we include it in our model.

Similar word Similar words (lemmas) to the target verb. Similar words are automatically obtained from a corpus (the same corpus as used for feature extraction) by Lin (1998)'s method. This feature is motivated by the hypothesis that distributionally similar words tend to be classified into the same class. Because Lin's method is based on the similarity of words in syntactic slots, the obtained similar words are expected to represent a verb class that share selectional preferences.

Slot class Semantic classes of the head words of the arguments. This feature is also intended to approximate selectional preferences. The semantic classes are obtained by clustering nouns, verbs, and adjectives into 200, 100, and 50 classes respectively, by using the k-medoid method with Lin (1998)'s similarity.

Figure 5 shows an example of the features for "sway," extracted from the BNC corpus.[3] Feature values are defined as relative frequencies for each lemma; while, for similar word features, feature values are weighted by Lin's similarity measure.

4.3 Tasks

We evaluate our model in the tasks of automatic verb classification (a.k.a. lexicon expansion): given gold verb-class associations for some set of verbs, we predict the classes for unknown

[3] "C82" and "C12" are automatically assigned cluster names.

Verb class	Levin class number
Recipient	13.1, 13.3
Admire	31.2
Amuse	31.1
Run	51.3.2
Sound Emission	43.2
Light and Substance Emission	43.1, 43.4
Cheat	10.6
Steal and *Remove*	10.5, 10.1
Wipe	10.4.1, 10.4.2
Spray/Load	9.7
Fill	9.8
Other Verbs of Putting	9.1–6
Change of State	45.1–4
Object Drop	26.1, 26.3, 26.7

Table 1: 14 classes used in Joanis et al. (2008) and their corresponding Levin class numbers

verbs. While our main target is the full set of VerbNet classes, we also show results for the task studied in the previous work.

14-class task The task to classify (almost) monosemous verbs into 14 classes. Refer to Table 1 for the definition of the 14 classes. Following Joanis et al. (2008)'s task definition, we removed verbs that belong to multiple classes in these 14 classes, and also removed *overly polysemous* verbs (in our experiment, verb-class associations that have the relative frequency that is less than 0.5 in SemLink are removed). For each class, member verbs are randomly split into 50% (training), 25% (development), and 25% (final test) sets.

All-class task The task to classify all target verbs into 268 classes.[4] Any verbs that did not occur at least 100 times in the BNC corpus were removed.[5] The remaining verbs (2517 words) are randomly split into 80% (training), 10% (development), and 10% (final test) sets, under the constraint that at least one instance for each class is included in the training set.[6]

4.4 Evaluation measures

For the 14-class task, we simply measure the classification accuracy. However, the evaluation in the all-class task is not trivial, because verbs may be assigned multiple classes.

Since our purpose is to obtain a probabilistic model rather than to classify monosemous verbs, the evaluation criterion should be sensitive to the probabilistic distribution on the test data. In this paper, we adopt two evaluation measures. One is the *top-N weighted accuracy*; we count the number of correct pairs $\langle c, v \rangle$ in the N-best outputs from the model (where N is the number of gold classes for each lemma), where each count is weighted by the relative frequency (i.e., the counts in SemLink) of the pair in the test set. For example, in the case for "blare" in Figure 4, if the model states that **Sound Emission** has the largest probability, we get 0.7 points. If **Manner Speaking** has the largest probability, we instead obtain 0.3 points. Intuitively, the score is higher when the model presents larger probabilities to classes with higher relative frequencies. This measure is similar to the top-N precision in information retrieval; it evaluates the ranked output by the model. It is intuitively interpretable, but is insufficient for evaluating the quality of probability distributions.

The other measure is *KL-divergence*, which is popularly used for measuring the dissimilarity between two probability distributions. This is defined as follows:

$$KL(p||q) = \sum_x p(x)\log(p(x)) - p(x)\log(q(x)).$$

In the experiments, this measure is applied, with the assumption that p is the relative frequency of $\langle c, v \rangle$ in the test set, and that q is the estimated probability distribution. Although the KL-divergence is not a true distance metric, it is sufficient for measuring the fitting of the estimated model to the true distribution. We report the KL-divergence averaged over all verbs in the test set. Since this measure indicates a dissimilarity, a smaller value is better. When p and q are equivalent, $KL(p||q) = 0$.

5 Experimental results

Table 2 shows the accuracy obtained for the 14-class task. The first column denotes the incorporated features ("Joanis et al.'s features" or "All features"), and the sources of the features ("WSJ" or "BNC"). The two baseline results are also given: "Baseline (*random*)" indicates that classes are randomly output, and "Baseline (*majority*)" indicates

[4] Two classes (**Being Dressed** and **Debone**) are not used in the experiments because no lemmas belonged to these classes after filtering by the frequency in BNC.

[5] This is the same preprocessing as Joanis et al. (2008), although we use VerbNet, while Joanis et al. (2008) used the original Levin classifications.

[6] Because polysemous verbs belong to multiple classes, the class-wise data split was not adopted for the all-class task.

	Accuracy
Baseline (*random*)	7.14
Baseline (*majority*)	26.47
Joanis et al.'s features/WSJ	56.86
Joanis et al.'s features/BNC	64.22
All features/WSJ	60.29
All features/BNC	68.14

Table 2: Accuracy for the 14-class task

	Accuracy	KL
Baseline (*random*)	0.37	—
Baseline (*majority*)	8.69	—
Joanis et al.'s features/WSJ	30.26	3.65
Joanis et al.'s features/BNC	35.66	3.32
All features/WSJ	34.07	3.37
All features/BNC	42.54	2.99

Table 3: Accuracy and KL-divergence for the all-class task (the VerbNet+SemLink setting)

that the majority class (i.e., the class that has the largest number of member verbs) is output to every lemma. While these figures cannot be compared directly to the previous works due to the difference in the preprocessing, Joanis et al. (2008) achieved 58.4% accuracy for the 14-class task. Table 3 and 4 present the results for the all-class task. Table 3 gives the accuracy and KL-divergence achieved by the model trained with the VerbNet+SemLink training instances, while Table 4 presents the same measures by the training instances created from VerbNet only.

Our models performed substantially better on both tasks than the baseline models. The results also proved that the features we proposed in this paper contributed to the further improvement of the model from Joanis et al. (2008). In the all-class task with the VerbNet+SemLink setting, our features achieved 10.69% error reduction in the accuracy over Joanis et al. (2008)'s features. Another interesting fact is that the model with BNC consistently outperformed the model with WSJ. This outcome is somewhat surprising, provided that the relative frequencies in the training/test sets are created from the WSJ portion of SemLink. The reason for this is independent of the corpus size, as will be shown below. When comparing Table 3 and 4, we can see that using SemLink statistics resulted in a slightly better model. This result is predictable, because the evaluation measures are sensitive to the relative frequencies estimated from SemLink. However, the difference remained small. In both of the tasks and the evaluation measures, the best model was achieved when we use

	Accuracy	KL
Baseline (*random*)	0.37	—
Baseline (*majority*)	8.69	—
Joanis et al.'s features/WSJ	29.65	3.67
Joanis et al.'s features/BNC	35.78	3.34
All features/WSJ	34.53	3.40
All features/BNC	42.38	3.02

Table 4: Accuracy and KL-divergence for the all-class task (the VerbNet only setting)

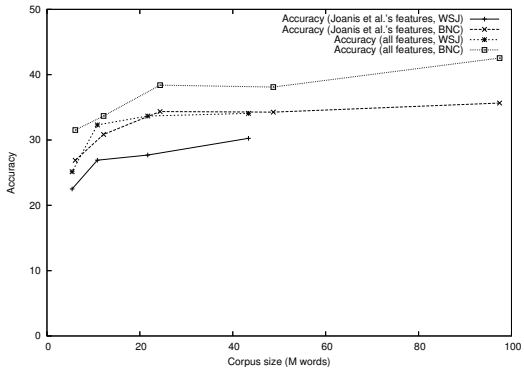


Figure 6: Corpus size vs. accuracy

all the features extracted from BNC, and create training instances from VerbNet+SemLink.

Figure 6 and 7 plot the accuracy and KL-divergence against the size of the unannotated corpus used for feature extraction. The result clearly indicates that the learning curve still grows at the corpus size with 100 million words (especially for the all features + BNC setting), which indicates that better models are obtained by increasing the size of the unannotated corpora.

Therefore, we can claim that the differences between the domains and the size of the unannotated corpora are more influential than the availability of the annotated corpora. This indicates that learning only from a lexicon would be a viable solution, when a token-disambiguated corpus like SemLink is unavailable.

Table 5 shows the contribution of each feature group. BNC is used for feature extraction, and VerbNet+SemLink is used for the creation of training instances. The results demonstrated the effectiveness of the slot POS features, and in particular, for the all-class task, most likely because VerbNet covers verbs that take non-nominal arguments. Additionally, the similar word features contributed equally or more in both of the tasks. This result suggests that we were reasonable in hypothesizing that distributionally similar words tend to be clas-

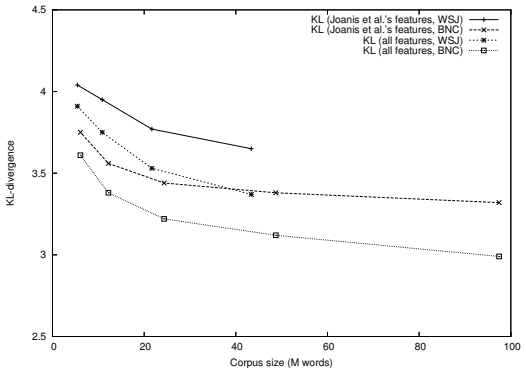


Figure 7: Corpus size vs. KL-divergence

	14-classes Accuracy	All classes Accuracy	KL
Baseline (*random*)	7.14	0.37	—
Baseline (*majority*)	26.47	8.69	—
Joanis et al.'s features	64.22	35.66	3.32
+ Slot POS	66.67	38.77	3.18
+ Syntactic frame	64.71	35.99	3.29
+ Similar word	68.14	37.88	3.10
+ Slot class	64.71	36.51	3.26
All features	68.14	42.54	2.99

Table 5: Contribution of features

sified into the same class. Slot classes also contributed to a slight improvement, indicating that selectional preferences are effective clues for predicting VerbNet classes. The result of the "All features" model for the all-class task attests that these features worked collaboratively, and using them all resulted in a considerably better model.

From the analysis of the confusion matrix for the outputs by our best model, we identified several reasons for the remaining misclassification errors. A major portion of the errors were caused by confusing the classes that take the same prepositions. Examples of these errors include:

- **Other Change of State** verbs were misclassified into the ***Butter*** class: "embalm," "laminate." (they take "with" phrases)
- **Judgement** verbs were misclassified into the **Characterize** class: "acclaim," "hail." (they take "as" phrases)

Since prepositions are strong features for automatic verb classification (Joanis et al., 2008), the classes that take the same prepositions remained confusing. The discovery of the features to discriminate among these classes would be crucial for further improvement.

Another major error is in classifying verbs into **Other Change of State**. Examples include:

- ***Amuse*** verbs: "impair," "recharge."
- ***Herd*** verbs: "aggregate," "mass."

Because **Other Change of State** is one of the biggest classes, supervised learning tends to place a high probability to this class. Therefore, when strong clues do not exist, verbs tend to be misclassified into this class. In addition, this class is not syntactically/semantically homogeneous, and is likely to introduce noise in the machine learning classifier. A possible solution to this problem would be to exclude this class from the classification, and to process the class separately.

6 Conclusions

We presented a method for the supervised learning of a probabilistic model for a lexicon of VerbNet classes. By combining verb-class associations from VerbNet and SemLink, and features extracted from a large unannotated corpus, we could successfully train a log-linear model in a supervised way. The experimental results attested to our success that features proposed in this paper worked effectively in obtaining a better probability distribution. Not only syntactic features, but also semantic features were shown to be effective. While each of these features could increase the accuracy, they collaboratively contributed to a large improvement. In the all-class task, we obtained 10.69% error reduction in the classification accuracy over Joanis et al. (2008)'s model. We also observed the trend that a larger corpus for feature extraction led to a better model, indicating that a better model will be obtained by increasing the size of an unannotated corpus.

We could identify the effective features and settings for this problem, but the classification into all VerbNet classes remained challenging. One possible direction for this research topic would be to use our model for the semi-automatic construction of verb lexicons, with the help of human curation. However, there is also a demand for exploring other types of features that can discriminate among confusing classes.

Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research and Grant-in-Aid for Young Scientists (MEXT, Japan).

References

Omri Abend, Roi Reichart, and Ari Rappoport. 2008. A supervised algorithm for verb disambiguation into VerbNet classes. In *Proceedings of COLING 2008*, pages 9–16.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of COLING-ACL 1998*.

Bonnie J. Dorr and Doug Jones. 1996. Role of word sense disambiguation in lexical acquisition: Predicting semantics from syntactic cues. In *Proceedings of COLING-96*, pages 322–327.

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

Roxana Girju, Dan Roth, and Mark Sammons. 2005. Token-level disambiguation of VerbNet classes. In *The Interdisciplinary Workshop on Verb Features and Verb Classes*.

Eric Joanis and Suzanne Stevenson. 2003. A general feature space for automatic verb classification. In *Proceedings of EACL 2003*, pages 163–170.

Eric Joanis, Suzanne Stevenson, and David James. 2008. A general feature space for automatic verb classification. *Natural Language Engineering*, 14(3):337–367.

Karin Kipper, Hoa Trang Dang, and Martha Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of 17th National Conference on Artificial Intelligence*.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2006. Extending VerbNet with novel verb classes. In *Proceedings of LREC 2006*.

Karin Kipper-Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, Computer and Information Science Department, University of Pennsylvania.

Anna Korhonen and Ted Briscoe. 2004. Extended lexical-semantic classification of English verbs. In *Proceedings of the HLT/NAACL Workshop on Computational Lexical Semantics*.

Anna Korhonen, Yuval Krymolowski, and Zvika Marx. 2003. Clustering polysemic subcategorization frame distributions semantically. In *Proceedings of ACL 2003*.

Anna Korhonen. 2002. Semantically motivated subcategorization acquisition. In *Proceedings of the Workshop on Unsupervised Lexical Acquisition*, pages 51–58.

Mirella Lapata and Chris Brew. 2004. Verb class disambiguation using informative priors. *Computational Linguistics*, 30(1):45–75.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago.

Juanguo Li and Chris Brew. 2007. Disambiguating Levin verbs using untagged data. In *Proceedings of RANLP 2007*.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL 1998*.

Edward Loper, Szu ting Yi, and Martha Palmer. 2007. Combining lexical resources: Mapping between PropBank and VerbNet. In *Proceedings of the 7th International Workshop on Computational Linguistics, Tilburg, the Netherlands*.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant senses in untagged text. In *Proceedings of ACL 2004*.

Diana McCarthy. 2001. *Lexical Acquisition at the Syntax-Semantics Interface: Diathesis Alternations, Subcategorization Frames and Selectional Preferences*. Ph.D. thesis, University of Sussex.

Paola Merlo and Suzanne Stevenson. 2001. Automatic verb-classification based on statistical distribution of argument structure. *Computational Linguistics*, 27(3):373–408.

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

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

Sabine Schulte im Walde. 2000. Clustering verbs semantically according to their alternation behavior. In *Proceedings of COLING 2000*, pages 747–753.

Sabine Schulte im Walde. 2003. Experiments on the choice of features for learning verb classes. In *Proceedings of EACL 2003*, pages 315–322.

Suzanne Stevenson and Eric Joanis. 2003. Semi-supervised verb class discovery using noisy features. In *Proceedings of CoNLL 2003*, pages 71–78.

Suzanne Stevenson and Paola Merlo. 1999. Automatic verb classification using grammatical features. In *Proceedings of EACL 1999*, pages 45–52.

Suzanne Stevenson, Paola Merlo, Natalia Kariaeva, and Kamin Whitehouse. 1999. Supervised learning of lexical semantic verb classes using frequency distributions. In *Proceedings of SigLex99: Standardizing Lexical Resources*, pages 15–22.

A Study on the Semantic Relatedness of Query and Document Terms in Information Retrieval

Christof Müller and **Iryna Gurevych**
Ubiquitous Knowledge Processing (UKP) Lab
Computer Science Department
Technische Universität Darmstadt, Hochschulstraße 10
D-64289 Darmstadt, Germany
http://www.ukp.tu-darmstadt.de/

Abstract

The use of lexical semantic knowledge in information retrieval has been a field of active study for a long time. Collaborative knowledge bases like Wikipedia and Wiktionary, which have been applied in computational methods only recently, offer new possibilities to enhance information retrieval. In order to find the most beneficial way to employ these resources, we analyze the lexical semantic relations that hold among query and document terms and compare how these relations are represented by a measure for semantic relatedness. We explore the potential of different indicators of document relevance that are based on semantic relatedness and compare the characteristics and performance of the knowledge bases Wikipedia, Wiktionary and WordNet.

1 Introduction

Today we face a rapidly growing number of electronic documents in all areas of life. This demands for more effective and efficient ways of searching these documents for information. Especially user-generated content on the web is a growing source of huge amounts of data that poses special difficulties to IR. The precise wording is often difficult to predict and current information retrieval (IR) systems are mainly based on the assumption that the meaning of a document can be inferred from the occurrence or absence of terms in it. In order to yield a good retrieval performance, i.e., retrieving all relevant documents without retrieving non-relevant documents, the query has to be formulated by the user in an appropriate way. Blair and Maron (1985) showed that with larger growing document collections, it gets impossible for the user to anticipate the terms that occur in all relevant documents, but not in non-relevant ones.

The use of semantic knowledge for improving IR by compensating non-optimal queries has been a field of study for a long time. First experiments on query expansion by Voorhees (1994) using lexical-semantic relations extracted from a linguistic knowledge base (LKB), namely WordNet (Fellbaum, 1998), showed sginificant improvements in performance only for manually selected expansion terms. The combination of WordNet with thesauri built from the underlying document collections by Mandala et al. (1998) improved the performance on several test collections. Mandala et al. (1998) identified missing relations, especially cross part of speech relations and insufficient lexical coverage as reasons for the low performance improvement when using only WordNet.

In recent work, collaborative knowledge bases (CKB) like Wikipedia have been used in IR for judging the document relevance by computing the semantic relatedness (SR) of queries and documents (Gurevych et al., 2007; Egozi et al., 2008; Müller and Gurevych, 2008) and have shown promising results. These resources have a high coverage of general and domain-specific terms. They are employed in several SR measures such as *Explicit Semantic Analysis* (ESA) (Gabrilovich and Markovitch, 2007) that allow the cross part of speech computation of SR and are not restricted to standard lexical semantic relations.

The goal of this paper is to shed light on the role of lexical semantics in IR and the way it can improve the performance of retrieval systems. There exist different kinds of resources for lexical semantic knowledge and different ways to embed this knowledge into IR. Wikipedia and Wiktionary, which have been applied in computational methods only recently, offer new possibilities to enhance IR. They have already shown an excellent performance in computing the SR of word pairs (Strube and Ponzetto, 2006; Gabrilovich and

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1338–1347,
Singapore, 6-7 August 2009.

Markovitch, 2007; Zesch et al., 2008). However, it is not yet clearly understood, what the most beneficial method is to employ SR using these resources in IR. We therefore perform a comparative study on an IR benchmark. We particularly analyze the contribution of SR of query and document terms to this task. To motivate those experiments we first prove that there exists a vocabulary gap in the test collection between queries and documents and show that the gap can be reduced by using lexical semantic knowledge. As the vocabulary coverage of knowledge bases is a crucial factor for being effective in IR, we compare the coverage of Wikipedia, Wiktionary and WordNet. We then analyze the lexical semantic relations that hold among query and document terms and how they are represented by the values of a SR measure. Finally, we explore the potential of different SR-based indicators of document relevance.

The remainder of this paper is structured as follows: In Section 2 we give a short overview of the LKBs and CKBs and the measure of SR we employ in this paper. The test collection we use in our experiments is described in Section 3. In Section 4 we analyze the vocabulary of the test collection and determine the coverage of the knowledge bases. This is followed by the examination of lexical semantic relations and the analysis of the SR of query terms in relevant and non-relevant documents in Section 5.

2 Knowledge Sources and Semantic Relatedness Measure

2.1 Linguistic Knowledge Bases

LKBs are mainly created by trained linguists following clearly defined guidelines. Therefore, their content is typically of high quality. This labor and cost intensive approach, however, yields a number of disadvantages for LKBs:

- their coverage and size are limited;
- they lack domain-specific vocabulary;
- continuous maintenance is often not feasible;
- the content can quickly be out-dated;
- only major languages are typically supported.

The most common types of LKBs are (i) dictionaries, which alphabetically list words and their senses of a certain language along with their definitions and possibly some additional information and (ii) thesauri, which group words with similar meaning together and define further semantic relations between the words, e.g., antonymy. The most widely used LKB is WordNet, which is a combination of dictionary and thesaurus. Since the hypernym and hyponym relations between noun groups form an *is-a* hierarchy, WordNet can also be seen as an ontology. The current version 3.0 of WordNet, which we use in our experiments, contains over 155,000 English words organized into almost 118,000 so called synsets, i.e., groups of synonymous words. WordNet covers mainly general vocabulary terms and its strongest part is the noun hierarchy.

2.2 Collaborative Knowledge Bases

Enabled by the development of Web 2.0 technology and created by communities of volunteers, CKBs have emerged as a new source of lexical semantic knowledge in recent years. In contrast to LKBs, they are created by persons with diverse personal backgrounds and fields of expertise. CKBs have the advantage of being freely available unlike many LKBs. However, the content of CKBs is mainly semi- or unstructured text which initially requires the extraction of explicit knowledge that can then be used in computational methods.

One of the CKBs we use in this paper is Wikipedia, a freely available encyclopedia. It currently contains more than 12 million articles in 265 languages. Besides articles, Wikipedia also offers other forms of knowledge that can be used in computational methods. This includes the hierarchy of article categories (Strube and Ponzetto, 2006; Zesch et al., 2007) and links between articles in the same language (Milne and Witten, 2008) and across languages (Schönhofen et al., 2007; Potthast et al., 2008; Sorg and Cimiano, 2008; Müller and Gurevych, 2008). Due to its encyclopedic character, Wikipedia contains many named entities and domain-specific terms which are not found in WordNet. In our experiments we used the Wikipedia dump of February 6th, 2007.

The second CKB we use is Wiktionary which is a multilingual dictionary and an affiliated project of Wikipedia. It resembles WordNet by containing synonym and hyponym information. It also contains information usually not found in LKBs like abbreviations, compounds, contractions, and the etymology of words. The 171 language-specific

editions of Wiktionary contain more than 5 million entries. Note that each language-specific edition contains not only entries for words of that particular language, but also for words of foreign languages. Wiktionary has been used in IR (Müller and Gurevych, 2008; Bernhard and Gurevych, 2009) and other tasks like sentiment analysis (Chesley et al., 2006) or ontology learning (Weber and Buitelaar, 2006). In our experiments we used the Wiktionary dump of Oct 16, 2007.

2.3 Semantic Relatedness Measure

A wide range of methods for measuring the SR of term pairs are discussed in the literature. In our experiments, we employ ESA as it can be used with all three knowledge bases in our experiments and has shown an excellent performance in related work. ESA was introduced by Gabrilovich and Markovitch (2007) employing Wikipedia as a knowledge base. Zesch et al. (2008) explored its performance using Wiktionary and WordNet as knowledge bases.

The idea of ESA is to express a term's meaning by computing its relation to Wikipedia articles. Each article title in Wikipedia is referred to as a concept and the article's text as the textual representation of this concept. A term is represented as a high dimensional concept vector where each value corresponds to the term's frequency in the respective Wikipedia article. The SR of two terms is then measured by computing the cosine between the respective concept vectors. When applying ESA to Wiktionary and WordNet, each word and synset entry, respectively, is referred to as a distinct concept, and the entry's information[1] is used as the textual representation of the concept.

In our experiments, we apply pruning methods as proposed by Gabrilovich and Markovitch (2007) with the goal of reducing noise and computational costs. Wikipedia concepts are not taken into account where the respective Wikipedia articles have less than 100 words or fewer than 5 in- or outlinks. For all three knowledge bases, concepts are removed from a term's concept vector if their normalized values are below a predefined threshold (empirically set to *0.01*).

[1]For WordNet, the glosses and example sentences of the synsets are used. Wiktionary does not contain glosses for all entries due to instance incompleteness. Therefore, a concatenation of selected information from each entry is used. See Zesch et al. (2008) for details.

Documents	
Number of documents	319115
Number of unique terms	400194
Ave. document length	256.23
Queries	
Number of queries	50
Number of unique terms	117
Ave. query length	2.44

Table 1: Statistics of the test data (after preprocessing).

3 Data

For our study we use parts of the data from the HARD track at the TREC 2003 conference[2]. The document collection consists of newswire text data in English from the year 1999, drawn from the Xinhua News Service (People's Republic of China), the New York Times News Service, and the Associated Press Worldstream News Service.[3] As we did not have access to the other document collections in the track, we restrict our experiments to the newswire text data.

From the 50 available topics of that track, we use only the title field, which consists of a few keywords describing the information need of a user. Table 1 shows some descriptive statistics of the documents and topics. The topics cover general themes like *animal protection*, *Y2K crisis* or *Academy Awards ceremony*. For the preprocessing of topics and documents we use tokenization, stopword removal and lemmatization employing the TreeTagger (Schmid, 1994). In our study, we rely on the relevance assessments performed at TREC to distinguish between relevant and non-relevant documents for each topic.

4 Vocabulary Mismatch

To confirm the intuition that there exists a vocabulary mismatch between queries and relevant documents, we computed the overlap of the terms in queries and relevant documents. The results are shown in the column *String-based* in Table 2. Averaged over all 50 topics, 35.5% of the relevant documents do contain all terms of the query, and 86.5% contain at least one of the query terms. However, this means that 13.5% of the relevant documents do not contain any query term and

[2]`http://trec.nist.gov/`

[3]AQUAINT Corpus, Linguistic Data Consortium (LDC) catalog number LDC2002T31

Measure	*String-based*	*SR-Wikipedia*		*SR-Wiktionary*		*SR-WordNet*	
Threshold		*0.0*	*0.05*	*0.0*	*0.05*	*0.0*	*0.05*
Ave. number of documents where all query terms are matched (in %)	35.5	91.2	72.2	82.6	65.2	74.8	50.8
Ave. number of documents where at least one query term is matched (in %)	86.5	100.0	99.1	97.7	97.2	94.9	92.9
Ave. number of query terms matched per document (in%)	55.8	95.6	84.0	87.0	76.8	79.2	65.7

Table 2: Statistics about the matching of the terms of queries and relevant documents.

cannot be retrieved by simple string-matching retrieval methods. In average, each relevant document matches 55.8% of the query terms. With an average query length of 2.44 (see Table 1), this means that in general, only one of two query terms occurs in the relevant documents which significantly lowers the probability of these documents to have a high ranking in the retrieval result.

In a second experiment, we proved the effectiveness of the SR measure and knowledge bases in reducing the vocabulary gap by counting the number of query terms that match the terms in the relevant documents as string or are semantically related to them. The results are shown in Table 2 for the different knowledge bases in the columns *SR-Wikipedia*, *SR-Wiktionary* and *SR-WordNet*. In order to analyse the performance of the SR measure when excluding very low SR values that might be caused by noise, we additionally applied a threshold of 0.05, i.e. only values above this threshold were taken into account. The SR values range between 0 and 1. However, the majority of SR values lie between 0 and 0.1.

Without threshold, using Wikipedia as knowledge base, in 91.2% of the relevant documents all query terms were matched. For Wiktionary with 82.6% and WordNet with 74.8% the number is lower, but still more than twice as high as for the string-based matching. Wikipedia matches in all relevant documents at least one query term. The average number of query terms matched per document is also increased for all three knowledge bases. Applying a threshold of 0.05, the values decrease, but are still above the ones for string-based matching.

The sufficient coverage of query and document terms is crucial for the effectiveness of knowledge bases in IR. It was found that LKBs do not necessarily provide a sufficient coverage (Mandala et al., 1998). Table 3 shows the amount of terms in queries and documents that are contained in Wikipedia, Wiktionary and WordNet. Wikipedia contains almost all query terms and also shows the best coverage for the document terms, followed by Wiktionary and WordNet. The values for all three knowledge bases are all higher than 80% except for the percentage of queries or documents where all terms are covered and the number of covered unique terms. The low percentage of covered unique document terms for even Wikipedia is mostly due to named entities, misspellings, identification codes and compounds.

	SR-Wikipedia	*SR-Wiktionary*	*SR-WordNet*
Queries			
Percentage of queries where all terms are covered	98.0	78.0	62.0
Percentage of covered terms	99.2	89.3	80.3
Percentage of covered unique terms	99.1	88.9	80.3
Ave. percentage of covered terms per query	99.6	89.2	80.1
Ave. percentage of covered unique terms per query	99.6	89.2	80.1
Documents			
Percentage of documents where all terms are covered	7.9	0.3	0.2
Percentage of covered terms	96.5	88.5	84.3
Percentage of covered unique terms	34.5	12.9	10.0
Ave. Percentage of terms covered per document	97.4	91.8	88.8
Ave. percentage of covered unique terms per document	96.3	88.0	83.6

Table 3: Statistics about the coverage of the knowledge bases.

Judging from the number of covered query and document terms alone, one would expect Wikipedia to yield a better performance when applied in IR than Wiktionary and especially WordNet. The higher coverage of Wikipedia is due to its nature of being an encyclopedia featuring arbitrarily long articles whereas entries in WordNet, and also Wiktionary, have a rather short length following specific guidelines. The high coverage alone is however not the only important factor for the effectiveness of a resource. It was shown by Zesch et al. (2008) that Wiktionary outperforms Wikipedia

in the task of ranking word pairs by their semantic relatedness when taking into account only word pairs that are covered by both resources.

5 Comparison of Semantic Relatedness in Relevant and Non-Relevant Documents

We have shown in Section 4 that a mismatch between the vocabulary of queries and relevant documents exists and that the SR measure and knowledge bases can be used to address this gap. In order to further study the SR of query and document terms with the goal to find SR-based indicators for document relevance, we created sets of relevant and non-relevant documents and compared their characteristic values concerning SR.

5.1 Document Selection

For analysing the impact of SR in the retrieval process, we compare relevant and non-relevant documents that were assigned similar relevance scores by a standard IR system. For the document selection we followed a method employed by Vechtomova et al. (2005). We created two sets of documents for each topic: one for relevant and one for non-relevant documents. We first retrieved up to 1000 documents for the topic using the BM25 IR model[4] (Spärck Jones et al., 2000) as implemented by Terrier[5]. The relevant retrieved documents constituted the first set. For the second set we selected for each relevant retrieved document a non-relevant document which had the closest score to the relevant document. After selecting an equal number of relevant and non-relevant documents, we computed the mean average and the standard deviation for the scores of each set. If there was a substantial difference between the values of more than 20%, the sets were rearranged by exchanging non-relevant documents or excluding pairs of relevant and non-relevant documents. If this was not possible, we excluded the corresponding topic from the experiments.

Table 4 shows the statistics for the final sets. From the original 50 topics, 13 were excluded for the above stated reasons or because no relevant documents were retrieved. The average length of about 345 terms for relevant documents is almost 40% larger than the length of non-relevant documents. Also the average number of query term instances is 6.93 in cotrast to 4.64 for non-relevant documents. The large difference of average document length and query term instances suggests a larger difference of the average relevance scores than 20%. However, in the BM25 model the relevance score is decreased with increasing document length and additional occurrences of a query term have little impact after three or four occurrences.

	Rel.	Nonrel.	Diff. (%)
Number of queries	37	37	0
Number of documents	1771	1771	0
Mean BM25 document score	6.388	6.239	2.39
Stdev BM25 document score	1.442	1.288	12.00
Ave. query length	2.32	2.32	0
Ave. document length	345.22	248.89	38.70
Ave. query term instances in documents	6.93	4.64	49.35

Table 4: Data characteristics that are independent of the chosen knowledge base and threshold.

5.2 Types of Lexical Semantic Relations

The most common classical lexical semantic relations between words are synonymy, hyponymy and a couple of others. In order to analyze the importance of these relations in the retrieval process, we automatically annotated the relations that hold between query and document terms using WordNet. Table 5 shows the percentage of lexical semantic relations between query and document terms (normalized by the number of query and document terms). The table also shows the coverage of the relations by the SR measure, i.e. the percentage of annotated relations for which the SR measure computed a value above 0 or the threshold 0.05, respectively. The percentage of relation types in general is higher for relevant documents. Cohyponymy and synonymy are by far the most frequently occurring relation types with up to almost 6%. Hypernyms and hyponyms have both a percentage of less than 1%. Holonymy and meronymy do almost not occur.

When applying no threshold, the SR measure covers up to 21% of the synonyms and cohyponyms and up to 12% of the hyper- and hyponyms in relevant documents. Using Wiktionary as knowledge base, the SR measure shows a better coverage than with Wikipedia. This is consistent with the findings in Zesch et al. (2008).

[4] We used the default values for the constants of the model ($k_1 = 1.2, b = 0.75$).

[5] http://ir.dcs.gla.ac.uk/terrier/

Relation Type	Percentage	SR-Wikipedia 0.0	SR-Wikipedia 0.05	SR-Wiktionary 0.0	SR-Wiktionary 0.05	SR-WordNet 0.0	SR-WordNet 0.05
Relevant Documents							
synonymy	3.61	17.81	13.13	18.33	13.78	15.28	12.18
hypernymy	0.86	8.57	2.30	12.18	3.02	11.69	2.26
hyponymy	0.88	5.72	1.28	6.33	1.67	6.54	1.02
cohyponymy	5.64	19.49	10.49	21.04	10.05	16.85	8.14
holonymy	0.02	0.61	0.17	0.74	0.17	0.53	0.00
meronymy	0.07	1.94	0.78	2.23	0.74	1.88	0.76
non-classical	—	58.80	6.62	23.22	3.13	12.77	2.56
Non-Relevant Documents							
synonymy	3.41	15.84	12.41	16.46	12.90	14.19	11.44
hypernymy	0.56	6.10	1.95	9.43	2.10	8.93	1.57
hyponymy	0.74	4.77	1.00	6.35	1.40	5.90	0.78
cohyponymy	5.42	17.42	9.91	19.23	9.71	15.38	7.66
holonymy	0.02	0.39	0.09	0.49	0.09	0.32	0.00
meronymy	0.10	1.88	0.55	1.84	0.65	1.57	0.57
non-classical	—	57.33	5.54	21.92	2.59	11.77	2.15

Table 5: Percentage of lexical semantic relations between query and document terms and their coverage by SR scores above threshold 0.00 and 0.05 in percent.

The reason for this is the method for constructing the textual representation of the concepts in the SR measure, where synonyms and other related words are concatenated. Also SR-WordNet outperforms Wikipedia for hypernymy and hyponymy. In contrast to Wiktionary, no direct information about related words is used to construct the textual representation of concepts. However, the very short and specific representations are built from glosses and examples which often contain hypernym-hyponym pairs. As WordNet is used for both, the automatic annotation of lexical semantic relations and the computation of SR values, its lower term coverage in general has not much impact on this experiment, as only the relations between terms contained in WordNet are annotated.

More than half of the SR values using Wikipedia are computed for term pairs which were not annotated with a classical relation. This is depicted in Table 5 as non-classical relation. These non-classical relations can be for example functional relations (*pencil* and *paper*) (Budanitsky and Hirst, 2006). However, as WordNet covers only a small part of the terms in the test collection, some of the SR values refered to as non-classical relations might actually be classical relations. For Wiktionary and WordNet, the number of non-classical relations is much lower, due to their smaller size and the way the textual representations of concepts are constructed. In general, the average number of SR scores for classical and non-classical relations are almost consistently higher for relevant documents which suggests that the comparison of SR scores could be beneficial in the IR process.

Relation Type	SR-Wikipedia 0.0	SR-Wikipedia 0.05	SR-Wiktionary 0.0	SR-Wiktionary 0.05	SR-WordNet 0.0	SR-WordNet 0.05
Relevant Documents						
synonymy	0.362	0.371	0.372	0.374	0.366	0.368
hypernymy	0.021	0.021	0.021	0.019	0.016	0.019
hyponymy	0.017	0.021	0.008	0.012	0.007	0.015
cohyponymy	0.270	0.334	0.315	0.353	0.312	0.363
holonymy	0.001	0.000	0.001	0.000	0.000	0.000
meronymy	0.004	0.003	0.003	0.002	0.003	0.002
non-classical	0.045	0.356	0.098	0.491	0.205	0.599
Non-Relevant Documents						
synonymy	0.344	0.348	0.349	0.350	0.343	0.344
hypernymy	0.027	0.030	0.025	0.029	0.022	0.028
hyponymy	0.019	0.029	0.012	0.023	0.009	0.025
cohyponymy	0.250	0.295	0.277	0.312	0.295	0.334
holonymy	0.001	0.000	0.000	0.000	0.000	0.000
meronymy	0.003	0.002	0.002	0.002	0.003	0.002
non-classical	0.041	0.374	0.103	0.538	0.222	0.643

Table 6: Average values of SR scores corresponding to lexical semantic relations between query and document terms above threshold 0.00 and 0.05 in percent.

When applying a threshold of 0.05, the most visible effect is that the percentage of non-classical relations is decreasing much stronger than the percentage of classical relations. The comparison of the average SR values for each relation type in Table 6 confirms that this is due to the fact that the SR measure assigns on average higher values to the classical relations than to the non-classical relations. After applying a threshold of 0.05 the average SR values corresponding to non-classical relations increase and are equal to or higher than the values for classical relations. The values for classical relations are in general higher for relevant documents, whereas the values for non-classical relations are lower.

5.3 SR-based Indicators for Document Relevance

For each topic and document in one of the sets we computed the SR between the query and document terms. We then computed the arithmetic mean of the following characteristic values for each set: the sum of SR scores, the number of SR scores, the number of terms which are semantically related to a query term and the average SR score. In order to eliminate the difference in document length and average number of query term instances between the relevant and non-relevant sets, we normalized all values, except for the average SR score, by the document length and excluded the SR scores of query term instances.

Figure 1 shows the average difference of these values between relevant and non-relevant document sets for SR-thresholds from 0 to 0.6 (step-size=0.01). As the majority of the SR scores have a low value, there is not much change for thresholds above 0.5.

Except for the average SR score, the differences have a peak at thresholds between 0.01 and 0.09 and decrease afterwards to a constant value. The SR scores computed using Wikipedia show the highest differences. Wiktionary and WordNet perform almost equally, but show lower differences than Wikipedia, especially for the sum of scores. All three knowledge bases show higher differences for the number of scores and number of related terms than for the sum of scores. The differences at the peaks are statistically significant[6], except for the differences of the sum of scores for Wiktionary and WordNet.

For the average SR score, the differences are mostly negative at low thresholds and increase to a low positive value for higher thresholds. A higher number of very low SR values is computed for the relevant documents, which causes the average SR score to be lower than for the non-relevant documents at low thresholds.

Additionally, Figure 2 shows the percentage of topics where the mean value of the relevant document set is higher than the one of the non-relevant document set. Wikipedia shows the highest percentage with about 86% for the number of scores and the number of related terms. Wiktionary and WordNet have a low percentage for the sum of scores, but reach up to 75% for the number of scores and the number of related terms.

[6]We used the Wilcoxon test at a significance level of 0.05.

The analysis of the SR of query and document terms shows that there are significant differences for relevant and non-relevant documents that can be measured by computing SR scores with any of the three knowledge bases. Especially when using Wiktionary and WordNet, the number of SR scores and the number of related terms might be better indicators for the document relevance than the sum of SR scores.

6 Conclusions

The vocabulary mismatch of queries and documents is a common problem in IR, which becomes even more serious the larger the document collection grows. CKBs like Wikipedia and Wiktionary, which have been applied in computational methods only recently, offer new possibilities to tackle this problem. In order to find the most beneficial way to employ these resources, we studied the semantic relatedness of query and document terms of an IR benchmark and compared the characteristics and performance of the CKBs Wikipedia and Wiktionary to the LKB WordNet.

We first proved that there exists a vocabulary gap in the test collection between queries and documents and that it can be reduced by employing a concept vector based measure for SR with any of the three knowledge bases. Using WordNet to automatically annotate the lexical semantic relations of query and document terms, we found that cohyponymy and synonymy are the most frequent classical relation types. Although the percentage of annotated relations for which also the SR measure computed values above a predefined threshold was at best 21%, the average number of SR scores for classical and non-classical relations were almost consistently higher for relevant documents.

Comparing the number and the value of SR scores of query and document terms, a significant difference between relevant and non-relevant documents was observed by using any of the three knowledge bases. Although Wikipedia had the best coverage of collection terms and showed the best perfomance in our experiments, Wiktionary and Wikipedia also seem to have a sufficient size for being beneficial in IR. In comparison to our previous work where the sum of SR scores was used as an indicator for document relevance (Müller and Gurevych, 2008), the results suggest that the number of SR scores and the number of related terms might show a better performance, es-

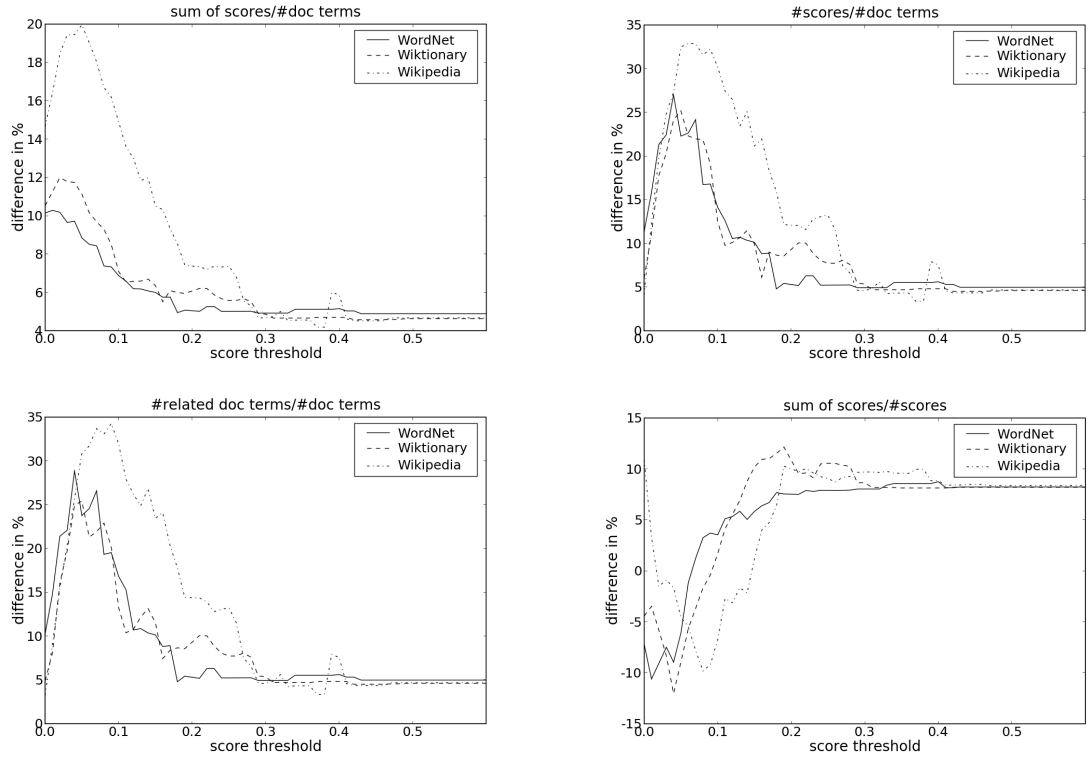


Figure 1: Differences between mean values of relevant and non-relevant document sets.

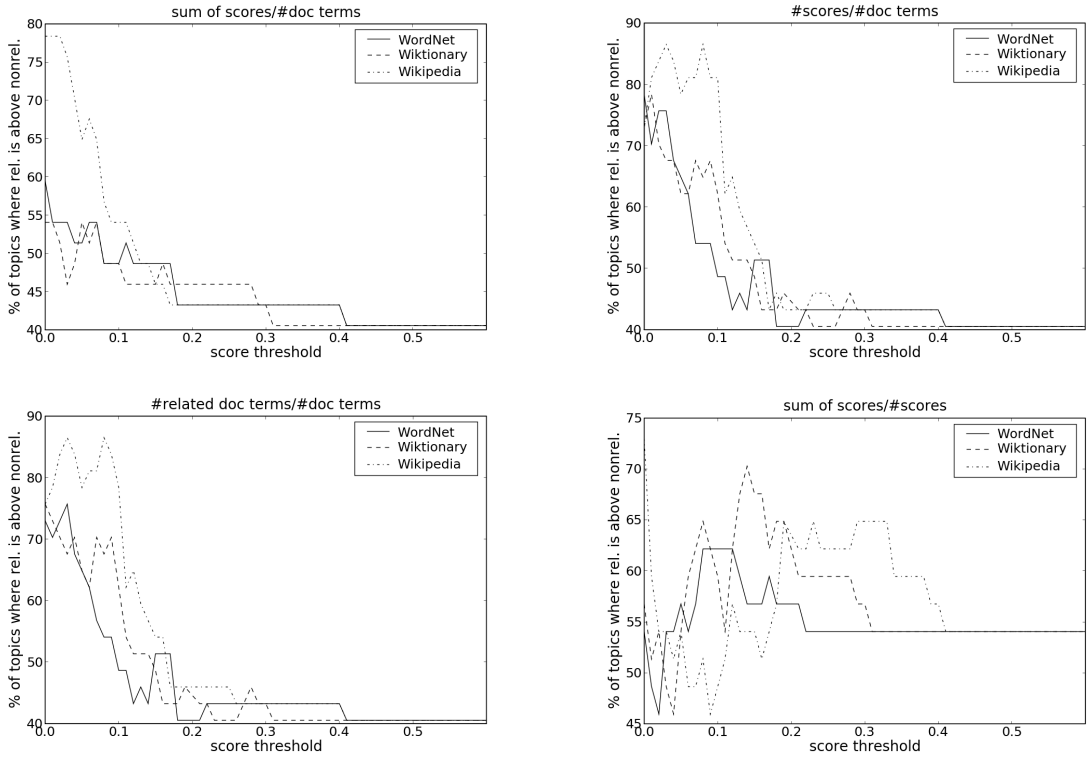


Figure 2: Percentage of topics where the mean value of the relevant document sets is higher than the one of the non-relevant document sets.

pecially for Wiktionary and WordNet.

In our future work, we plan to extend our analysis to other test collections and to query expansion methods in order to generalize our conclusions. As the problem of language ambiguity has a high impact on the use of SR measures, we will also consider word sense disambiguation in our future experiments.

Acknowledgments

This work was supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806 and by the German Research Foundation under grant No. GU 798/1-3. We would like to thank Aljoscha Burchardt for his helpful comments and the anonymous reviewers for valuable feedback on this paper.

References

D. Bernhard and I. Gurevych. 2009. Combining lexical semantic resources with question & answer archives for translation-based answer finding. In *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, Singapore, Aug.

D. C. Blair and M. E. Maron. 1985. An evaluation of retrieval effectiveness for a full-text document-retrieval system. *Commun. ACM*, 28(3):289–299.

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

P. Chesley, B. Vincent, L. Xu, and R. Srihari. 2006. Using Verbs and Adjectives to Automatically Classify Blog Sentiment. In *Proceedings of AAAI-CAAW-06*.

O. Egozi, E. Gabrilovich, and S. Markovitch. 2008. Concept-Based Feature Generation and Selection for Information Retrieval. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, Chicago, IL.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

E. Gabrilovich and S. Markovitch. 2007. Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of The Twentieth International Joint Conference for Artificial Intelligence*, pages 1606–1611, Hyderabad, India.

I. Gurevych, C. Müller, and T. Zesch. 2007. What to be? - Electronic Career Guidance Based on Semantic Relatedness. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 1032–1039, Prague, Czech Republic, June.

R. Mandala, T. Tokunaga, and H. Tanaka. 1998. The Use of WordNet in Information Retrieval. In Sanda Harabagiu, editor, *Proceedings of the COLING-ACL workshop on Usage of WordNet in Natural Language Processing*, pages 31–37. Association for Computational Linguistics, Somerset, New Jersey.

D. Milne and I. Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Wikipedia and AI workshop at the AAAI-08 Conference (WikiAI08)*, Chicago, USA.

C. Müller and I. Gurevych. 2008. Using Wikipedia and Wiktionary in Domain-Specific Information Retrieval. In F. Borri, A. Nardi, and C. Peters, editors, *Working Notes for the CLEF 2008 Workshop*, Aarhus, Denmark, Sep.

M. Potthast, B. Stein, and M. Anderka. 2008. A Wikipedia-Based Multilingual Retrieval Model. In C. Macdonald, I. Ounis, V. Plachouras, I. Ruthven, and R. W. White, editors, *30th European Conference on IR Research, ECIR 2008, Glasgow*, volume 4956 of *LNCS*, pages 522–530. Springer.

H. Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of Conference on New Methods in Language Processing*.

P. Schönhofen, I. Biro, A. A. Benczur, and K. Csalogany. 2007. Performing Cross Language Retrieval with Wikipedia. In *Working Notes for the CLEF 2007 Workshop*.

P. Sorg and P. Cimiano. 2008. Cross-lingual Information Retrieval with Explicit Semantic Analysis. In F. Borri, A. Nardi, and C. Peters, editors, *Working Notes for the CLEF 2008 Workshop*, Aarhus, Denmark, Sep.

K. Spärck Jones, S. Walker, and S. E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management*, 36(6):779–808 (Part 1); 809–840 (Part 2).

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

O. Vechtomova, M. Karamuftuoglu, and S. E. Robertson. 2005. A Study of Document Relevance and Lexical Cohesion between Query Terms. In *Proceedings of the Workshop on Methodologies and Evaluation of Lexical Cohesion Techniques in Real-World Applications (ELECTRA 2005), the 28th Annual International ACM SIGIR Conference*, pages 18–25, Salvador, Brazil, August.

E. M. Voorhees. 1994. Query expansion using lexical-semantic relations. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 61–69, New York, NY, USA. Springer-Verlag New York, Inc.

N. Weber and P. Buitelaar. 2006. Web-based Ontology Learning with ISOLDE. In *Proc. of the Workshop on Web Content Mining with Human Language at the International Semantic Web Conference*, Athens GA, USA, 11.

T. Zesch, I. Gurevych, and M. Mühlhäuser. 2007. Comparing Wikipedia and German Wordnet by Evaluating Semantic Relatedness on Multiple Datasets. In *Proceedings of HLT-NAACL*, pages 205–208.

T. Zesch, C. Müller, and I. Gurevych. 2008. Using Wiktionary for Computing Semantic Relatedness. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*, pages (861–867), Chicago, Illinois, USA.

Predicting Subjectivity in Multimodal Conversations

Gabriel Murray and **Giuseppe Carenini**
University of British Columbia
Vancouver, Canada
(gabrielm, carenini)@cs.ubc.ca

Abstract

In this research we aim to detect subjective sentences in multimodal conversations. We introduce a novel technique wherein subjective patterns are learned from both labeled and unlabeled data, using n-gram word sequences with varying levels of lexical instantiation. Applying this technique to meeting speech and email conversations, we gain significant improvement over state-of-the-art approaches. Furthermore, we show that coupling the pattern-based approach with features that capture characteristics of general conversation structure yields additional improvement.

1 Introduction

Conversations are rich in subjectivity. Conversation participants agree and disagree with one other, argue for and against various proposals, and generally take turns expressing their private states. Being able to separate these subjective utterances from more objective utterances would greatly facilitate the analysis, mining and summarization of a large number of conversations.

Two of the most prevalent conversational media are meetings and emails. Face-to-face meetings enable numerous people to exchange a large amount of information and opinions in a short period of time, while emails allow for concise exchanges between potentially far-flung participants. Meetings and emails can also feed into one another, with face-to-face meetings occurring at regular intervals and emails continuing the conversations in the interim. This poses several interesting questions, such as whether subjective utterances are more or less likely to be found in email exchanges compared with meetings, and whether the ratios of positive and negative subjective utterances differ between the two modalities.

In this paper we describe a novel approach for predicting subjectivity, and test it in two sets of experiments on meetings and emails. Our approach combines a new general purpose method for learning subjective patterns, with features that capture basic characteristics of conversation structure across modalities. The subjective patterns are essentially n-gram sequences with varying levels of lexical instantiation, and we demonstrate how they can be learned from both labeled and unlabeled data. The conversation features capture structural characteristics of multimodal conversations as well as participant information.

We test our approach in two sets of experiments. The goal of the first set of experiments is to discriminate subjective from non-subjective utterances, comparing the novel approach to existing state-of-the-art techniques. In the second set of experiments, the goal is to discriminate positive-subjective and negative-subjective utterances, establishing their polarity. In both sets of experiments, we assess the impact of features relating to conversation structure.

2 Related Research

Raaijmakers et al. (2008) have approached the problem of detecting subjectivity in meeting speech by using a variety of multimodal features such as prosodic features, word n-grams, character n-grams and phoneme n-grams. For subjectivity detection, they found that a combination of all features was best, while prosodic features were less useful for discriminating between positive and negative utterances. They found character n-grams to be particularly useful.

Riloff and Wiebe (2004) presented a method for learning subjective extraction patterns from a large amount of data, which takes subjective and non-subjective text as input, and outputs significant lexico-syntactic patterns. These patterns are based on syntactic structure output by the Sundance shal-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1348–1357,
Singapore, 6-7 August 2009.

low dependency parser (Riloff and Phillips, 2004). They are extracted by exhaustively applying syntactic templates such as $< subj >$ *passive-verb* and *active-verb* $< dobj >$ to a training corpus, with an extracted pattern for every instantiation of the syntactic template. These patterns are scored according to probability of relevance given the pattern and frequency of the pattern. Because these patterns are based on syntactic structure, they can represent subjective expressions that are not fixed word sequences and would therefore be missed by a simple n-gram approach.

Riloff et al. (2006) explore feature subsumption for opinion detection, where a given feature may subsume another feature representationally if the strings matched by the first feature include all of the strings matched by the second feature. To give their own example, the unigram *happy* subsumes the bigram *very happy*. The first feature will *behaviorally* subsume the second if it representationally subsumes the second and has roughly the same information gain, within an acceptable margin. They show that they can improve opinion analysis results by modeling these relations and reducing the feature set.

Our approach for learning subjective patterns like Raaijmakers et al. relies on n-grams, but like Riloff et al. moves beyond fixed sequences of words by varying levels of lexical instantiation.

Yu and Hatzivassiloglou (2003) addressed three challenges in the news article domain: discriminating between objective documents and subjective documents such as editorials, detecting subjectivity at the sentence level, and determining polarity at the sentence level. They found that the latter two tasks were substantially more difficult than classification at the document level. Of particular relevance here is that they found that part-of-speech (POS) features were especially useful for assigning polarity scores, with adjectives, adverbs and verbs comprising the best set of POS tags. This work inspired us to look at generalization of n-grams based on POS.

On the slightly different task of classifying the intensity of opinions, Wilson et al. (2006) employed several types of features including dependency structures in which words can be backed off to POS tags. They found that this feature class improved the overall accuracy of their system.

Somasundaran et al. (2007) investigated subjectivity classification in meetings. Their findings indicate that both lexical features (list of words and expressions) and discourse features (dialogue acts and adjacency pairs) can be beneficial. In the same spirit, we effectively combine lexical patterns and conversational features.

The approach to predicting subjectivity we present in this paper is a novel contribution to the field of opinion and sentiment analysis. Pang and Lee (2008) give an overview of the state of the art, discussing motivation, features, approaches and available resources.

3 Subjectivity Detection

In this section we describe our approach to subjectivity detection. We begin by describing how to learn subjective n-gram patterns with varying levels of lexical instantiation. We then describe a set of features characterizing multimodal conversation structure which can be used to supplement the n-gram approach. Finally, we describe the baseline subjectivity detection approaches used for comparison.

3.1 Partially Instantiated N-Grams

Our approach to subjectivity detection and polarity detection is to learn significant patterns that correlate with the subjective and polar utterances. These patterns are word trigrams, but with varying levels of lexical instantiation, so that each unit of the n-gram can be either a word or the word's part-of-speech (POS) tag. This contrasts, then, with work such as that of Raaijmakers et al. (2008) who include trigram features in their experiments, but where their learned trigrams are fully instantiated. As an example, while they may learn that a trigram *really great idea* is positive, we may additionally find that *really great NN* and *RB great NN* are informative patterns, and these patterns may sometimes be better cues than the fully instantiated trigrams. To differentiate this approach from the typical use of trigrams, we will refer to it as the VIN (*varying instantiation n-grams*) method.

In some respects, our approach to subjectivity detection is similar to Riloff and Wiebe's work cited above, in the sense that their extraction patterns are partly instantiated. However, the AutoSlog-TS approach relies on deriving syntactic structure with the Sundance shallow parser (Riloff and Phillips, 2004). We hypothesize that our trigram approach may be more robust to disfluent and fragmented meeting speech and emails

1	2	3
really	great	idea
really	great	NN
really	JJ	idea
RB	great	idea
really	JJ	NN
RB	great	NN
RB	JJ	idea
RB	JJ	NN

Table 1: Sample Instantiation Set

on which syntactic parsers may perform poorly. Also, our learned trigram patterns range from fully instantiated to completely uninstantiated. For example, we might find that the pattern *RB JJ NN* is a very good indicator of subjective utterances because it matches a variety of scenarios where people are ascribing qualities to things, e.g. *really bad movie*, *horribly overcooked steak*. Notice that we do not see our approach and AutoSlog-TS as mutually exclusive, and indeed we demonstrate through these experiments that they can be effectively combined.

Our approach begins by running the Brill POS tagger (Brill, 1992) over all sentences in a document. We then extract all of the word trigrams from the document, and represent each trigram using every possible instantiation. Because we are working at the trigram level, and each unit of the trigram can be a word or its POS tag there are $2^3 = 8$ representations in each trigram's instantiation set. To continue the example from above, the instantiation set for the trigram *really great idea* is given in Table 1. As we scan down the instantiation set, we can see that the level of abstraction increases until it is completely uninstantiated. It is this multilevel abstraction that we are hypothesizing will be useful for learning new subjective and polar cues.

All trigrams are then scored according to their prevalence in relevant versus irrelevant documents (e.g. subjective vs. non-subjective sentences), following the scoring methodology of Riloff and Wiebe (2003). We calculate the conditional probability $p(relevance|trigram)$ using the actual trigram counts in relevant and irrelevant text. For learning negative-subjective patterns, we treat all negative sentences as the relevant text and the remainder of the sentences as irrelevant text, and conduct the same process for learning positive-subjective patterns. We consider significant patterns to be those where the conditional probability is greater than 0.65 and the pattern occurs more than five times in the entire document set (slightly higher than $probability >= 0.60$ and $frequency >= 2$ used by Riloff and Wiebe (2003)).

We possess a fairly small amount of conversational data annotated for subjectivity and polarity. The AMI meeting corpus and BC3 email corpus are described in more detail in Section 4.1. To address this shortfall in annotated data, we take two approaches to learning patterns. In the first, we learn a set of patterns from the annotated conversation data. In the second approach, we complement those patterns by learning additional patterns from unannotated data that are typically overwhelmingly subjective or objective in nature. We describe these two approaches here in turn.

3.1.1 Supervised Learning of Patterns from Conversation Data

The first learning strategy is to apply the above-described methods to the annotated conversation data, learning the positive patterns by comparing *positive-subjective* utterances to all other utterances, and learning the negative patterns by comparing the *negative-subjective* utterances to all other utterances, using the described methods. This results in 759 significant positive patterns and 67 significant negative patterns. This difference in pattern numbers can be explained by negative utterances being less common in the AMI meetings, as noted by Wilson (2008). It may be that people are less comfortable in expressing negative sentiments in face-to-face conversations, particularly when the meeting participants do not know each other well (in the AMI scenario meetings, many participants were meeting each other for the first time). But there may be a further explanation for why we learn many more positive than negative patterns. When conversation participants *do* express negative sentiments, they may couch those sentiments in more euphemistic or guarded terms compared with positive sentiments. Table 2 gives examples of significant positive and negative patterns learned from the labeled meeting data. The last two rows in Table 2 show how two patterns in the same instantiation set can have substantially different probabilities.

POS	$p(r\|t)$	NEG	$p(r\|t)$
you MD change	1.0	VBD not RB	1.0
should VBP DT	1.0	doesn't RB VB	0.875
very easy to	0.88	a bit JJ	0.66
we could VBP	0.78	think PRP might	0.66
NNS should VBP	0.71	be DT problem	0.71
PRP could do	0.66	doesn't really VB	0.833
it could VBP	83	doesn't RB VB	0.875

Table 2: Example Pos. and Neg. Patterns (AMI)

Pattern	$p(r\|t)$
can not VB	0.99
i can RB	0.99
i have not	0.98
do RB think	0.97
RB think that	0.95
RB agree with	0.95
IN PRP opinion	0.95

Table 3: Example Subjective Patterns (BLOG06)

3.1.2 Unsupervised Learning of Patterns from Blog Data

The second pattern learning strategy we take to learning subjective patterns is to use a relevant, but unannotated corpus. We focus on weblog (blog) data for several reasons. First, blog posts share many characteristics with both meetings and emails: they are conversational, informal and the language can be very ungrammatical. Second, blog posts are known for being subjective; bloggers post on issues that are passionate to them, offering arguments, opinions and invective. Third, there is a huge amount of available blog data. But because we do not possess blog data annotated for subjectivity, we take the following approach to learning subjective patterns from this data. We work on the assumption that a great many blog posts are inherently subjective, and that comparing this data to inherently *objective* text such as newswire articles, treating the latter as our irrelevant text, should lead to the detection of many new subjective patterns and greatly increase our coverage. While the patterns learned will be noisy, we hypothesize that the increased coverage will improve our subjectivity detection overall.

For our blog data, we use the BLOG06 Corpus[1] that was featured as training and testing data for the Text Analysis Conference (TAC) 2008 track on summarizing blog opinions. The portion used totals approximately 4,000 documents on all manner of topics. Treating that dataset as our relevant, subjective data, we then learn the subjective trigrams by comparing with the *irrelevant* TAC/DUC newswire data from the 2007 and 2008 update summarization tasks. To try to reduce the amount of noise in our learned patterns, we set the conditional probability threshold at 0.75 (vs. 0.65 for annotated data), and stipulate that all significant patterns must occur at least once in the irrelevant text. This last rule is meant to prevent us from learning completely blog-specific patterns such as *posted by NN* or *linked to DT*. In the end, more than 20,000 patterns were learned from the blog data. While manual inspection does show that many undesirable patterns were extracted, among the highest-scoring patterns are many sensible subjective trigrams such as those indicated in Table 3.

This approach is similar in spirit to the work of Biadsy et al. (2008) on unsupervised biography production. Without access to labeled biographical data, the authors chose to use sentences from Wikipedia biographies as their positive set and sentences from newswire articles as their negative set, on the assumption that most of the Wikipedia sentences would be relevant to biographies and most of the newswire sentences would not.

3.2 Deriving VIN Features

For our machine learning experiments, we derive, for each sentence, features indicating the presence of the significant VIN patterns. Patterns are binned according to their conditional probability range (i.e., $0.65 <= p < 0.75$, $0.75 <= p < 0.85$, $0.85 <= p < 0.95$, and $0.95 <= p$). There are three bins for the blog patterns, since the probability cutoff is 0.75 For each bin, there is a feature indicating the count of its patterns in the given sentence. When attempting to match these trigram patterns to sentences, we allow up to two wildcard lexical items between the trigram units. In this way a sentence can match a learned pattern even if the units of the n-gram are not contiguous (Raaijmakers et al. (2008) similarly include an n-gram feature allowing such intervening material).

A key reason for counting the number of matched patterns for each probability range as just described, rather than including a feature for each individual pattern, is to maintain the same level of dimensionality in our machine learning experiments when comparing the VIN approach to the baseline approaches described in Section 3.4.

[1] http://ir.dcs.gla.ac.uk/test_collections/blog06info.html

3.3 Conversational Features

While we hypothesize that the general purpose pattern-based approach described above will greatly aid subjectivity and polarity detection, we also recognize that there are many additional features specific for characterizing multimodal conversations that may correlate well with subjectivity and polarity. Such features include structural characteristics like the position of a sentence in a turn and the position of a turn in the conversation, and participant features relating to dominance or leadership. For example, it may be that subjective sentences are more likely to come at the end of a conversation, or that a person who dominates the conversation may utter more negative sentences.

We use the feature set provided by Murray and Carenini (2008), which they used for automatic summarization of conversations and which are shown in Table 4. Many of the features are based on so-called $Sprob$ and $Tprob$ term-weights, the former of which weights words based on their distributions across conversation participants and the latter of which similarly weights words based on their distributions across conversation turns. Other features include word entropy of the candidate sentence, lexical cohesion of the sentence with the greater conversation, and structural features indicating position of the candidate sentence in the turn and in the conversation, such as the elapsed time since the beginning of the conversation.

Feature ID	Description
MXS	max *Sprob* score
MNS	mean *Sprob* score
SMS	sum of *Sprob* scores
MXT	max *Tprob* score
MNT	mean *Tprob* score
SMT	sum of *Tprob* scores
TLOC	position in turn
CLOC	position in conv.
SLEN	word count, globally normalized
SLEN2	word count, locally normalized
TPOS1	time from beg. of conv. to turn
TPOS2	time from turn to end of conv.
DOM	participant dominance in words
COS1	cosine of conv. splits, w/ *Sprob*
COS2	cosine of conv. splits, w/ *Tprob*
PENT	entropy of conv. up to sentence
SENT	entropy of conv. after the sentence
THISENT	entropy of current sentence
PPAU	time btwn. current and prior turn
SPAU	time btwn. current and next turn
BEGAUTH	is first participant (0/1)
CWS	rough ClueWordScore (cohesion)
CENT1	cos. of sentence & conv., w/ *Sprob*
CENT2	cos. of sentence & conv., w/ *Tprob*

Table 4: Features Key

3.4 Baseline Approaches

There are two baselines in particular to which we are interested in comparing the VIN approach. As stated earlier, we are hypothesizing that the increasing levels of abstraction found with partially instantiated trigrams will lead to improved classification compared with using only fully instantiated trigrams. To test this, we also run the subjective/non-subjective and positive/negative experiments using *only* fully instantiated trigrams. There are 71 such positive trigrams and 5 such negative trigrams learned from the AMI data. There are just over 1200 fully instantiated trigrams learned from the unannotated BLOG06 data.

Believing that the current approach may offer benefits over state-of-the-art pattern-based subjectivity detection, we also implement the AutoSlog-TS method of Riloff and Wiebe (2003) for extracting subjective extraction patterns. In AutoSlog-TS, once all of the patterns are extracted using the Sundance parser, the scoring methodology is much the same as desribed in Section 3.1. Conditional probabilities are calculated by comparing pattern occurrences in the relevant text with occurrences in all text, and we again use a threshold of $p >= 0.65$ and $frequency >= 5$ for significant patterns. For the BLOG06 data, we use a probability cutoff of 0.75 as before. For deriving the features used in our machine learning experiments, the patterns are similarly grouped according to conditional probability. From the annotated data, 48 patterns are learned in total, 46 positive and only 2 negative. From the BLOG06 data, more than 3000 significant patterns are learned. Among significant patterns learned from the AMI corpus are $< subj >$ *BE good*, *change* $< dobj >$, $< subj >$ *agree* and *problem with* $< NP >$.

To gauge the effectiveness of the various feature types, for both sets of experiments we build multiple models on a variety of feature combinations: fully instantiated trigrams (TRIG), varying instantiation n-grams (VIN), AutoSlog-TS (SLOG), conversational structure features (CONV), and the set of all features.

4 Experimental Setup

In this section we describe the corpora used, the relevant subjectivity annotation, and the statistical

classifiers employed.

4.1 Corpora

We use two annotated corpora for these experiments. The AMI corpus (Carletta et al., 2005) consists of meetings in which participants take part in role-playing exercises concerning the design and development of a remote control. Participants are grouped in fours, and each group takes part in a sequence of four meetings, bringing the remote control from design to market. The four members of the group are assigned roles of project manager, industrial designer, user interface designer, and marketing expert. In total there are 140 such scenario meetings, with individual meetings ranging from approximately 15 to 45 minutes.

The BC3 corpus (Ulrich et al., 2008) contains email threads from the World Wide Web Consortium (W3C) mailing list. The threads feature a variety of topics such as web accessibility and planning face-to-face meetings. The annotated portion of the mailing list consists of 40 threads.

4.2 Subjectivity Annotation

Wilson (2008) has annotated 20 AMI meetings for a variety of subjective phenomena which fall into the broad classes of *subjective utterances*, *objective polar utterances* and *subjective questions*. It is this first class in which we are primarily interested here. Two subclasses of subjective utterances are *positive subjective* and *negative subjective* utterances. Such subjective utterances involve the expression of a private state, such as a positive/negative opinion, positive/negative argument, and agreement/disagreement. The 20 meetings were labeled by a single annotator, though Wilson (2008) did conduct a study of annotator agreement on two meetings, reporting a κ of 0.56 for detecting subjective utterances. Of the roughly 20,000 dialogue acts total in the 20 AMI meetings, nearly 4000 are labeled as *positive-subjective* and nearly 1300 as *negative-subjective*. For the first experimental task, we consider the subjective class to be the union of positive-subjective and negative-subjective dialogue acts. For the second experimental task, the goal is to discriminate positive-subjective from negative-subjective.

For the BC3 emails, annotators were initially asked to create extractive and abstractive summaries of each thread, in addition to labeling a variety of sentence-level phenomena, including whether each sentence was subjective. In a second round of annotations, three different annotators were asked to go through all of the sentences previously labeled as subjective and indicate whether each sentence was *positive*, *negative*, *positive-negative*, or *other*. The definitions for positive and negative subjectivity mirrored those given by Wilson (2008). For the purpose of these experiments, we consider a sentence to be subjective if at least two of the annotators labeled it as subjective, and similarly consider a subjective sentence to be positive or negative if at least two annotators label it as such. Using this majority vote labeling, 172 of 1800 sentences are considered subjective, with 44% of those labeled as *positive-subjective* and 37% as *negative-subjective*, showing that there is much more of a balance between positive and negative sentiment in these email threads compared with meeting speech (note that some subjective sentences are not positive or negative). The κ for labeling subjective sentences in the email corpus is 0.32. The lower annotator agreement on emails compared with meetings suggests that subjectivity in email text may be manifested more subtly or conveyed somewhat amibiguously.

4.3 Classifier and Experimental Setup

For these experiments we use a maximum entropy classifier using the *liblinear* toolkit[2] (Fan et al., 2008). Feature subset selection is carried out by calculating the F-statistic for each feature, ranking the features according to the statistic, and training on increasingly smaller subsets of feature in a cross-validation procedure, ultimately choosing the feature set with the highest balanced accuracy during cross-validation.

Because the annotated portions of our corpora are fairly small (20 meetings, 40 email threads), we employ a leave-one-out method for training and testing rather than using dedicated training and test sets. For the polarity labeling task applied to the BC3 corpus, we pool all of the sentences and perform 10-fold cross-validation at the sentence level.

4.4 Evaluation Metrics

We employ two sets of metrics for evaluating all classifiers: precision/recall/f-measure and the receiver operator characteristic (ROC) curve. The ROC curve plots the true-positive/false-positive ratio while the posterior threshold is varied, and

[2] http://www.csie.ntu.edu.tw/ cjlin/liblinear/

we report the area under the curve (AUROC) as the measure of interest. Random performance would feature an AUROC of approximately 0.5, while perfect classification would yield an AUROC of 1. The advantage of the AUROC score compared with precision/recall/f-measure is that it evaluates a given classifier across all thresholds, indicating the classifier's overall discriminating power. This metric is also known to be appropriate when class distributions are skewed (Fawcett, 2003), as is our case. For completeness we report both AUROC and p/r/f, but our discussions focus primarily on the AUROC comparisons.

5 Results

In this section we describe the experimental results, first for the subjective/non-subjective classification task, and subsequently for the positive-negative classification task.

5.1 Subjective / Non-Subjective Classification

For the subjectivity detection task, the results on the AMI and BC3 data closely mirrored each other, with the VIN approach constituting a very effective feature set, outperforming both baselines. We report the results on meeting and emails in turn.

5.1.1 AMI corpus

For the subjectivity task with the AMI corpus, we first report the precision, recall and f-measure results in Table 5 where the various classifiers are compared with a lower bound (LB) in which the positive class is always predicted, leading to perfect recall. It can be seen that the novel systems exhibit substantial improvement in precision and f-measure over this lower-bound. While the VIN approach yields the best precision scores, the full feature set achieves the highest f-measure.

As shown in Figure 1, the average AUROC with the VIN approach is 0.69, compared with 0.61 for AutoSlog-TS, a significant difference according to paired t-test ($p<0.01$). The VIN approach is also significantly better than the standard fully instantiated trigram pattern approach ($p<0.01$). This latter result suggests that the increased level of abstraction found in the varying instantiation n-grams does improve performance.

The conversational features alone give comparable performance to the VIN method (no significant difference), and the best results are found using the full feature set, which gives an average AUROC of 0.71, a significant improvement over VIN only ($p<0.05$).

Sys	Precision	Recall	F-Measure
AMI Corpus			
LB	26	100	41
Trig	25	63	36
Slog	39	48	43
VIN	41	58	48
Conv	36	73	49
All Feas	38	70	49
BC3 Corpus			
LB	10	100	17
Trig	27	10	14
Slog	24	13	17
VIN	27	22	24
Conv	25	29	27
All Feas	33	34	33

Table 5: P/R/F Results, Subjectivity Task

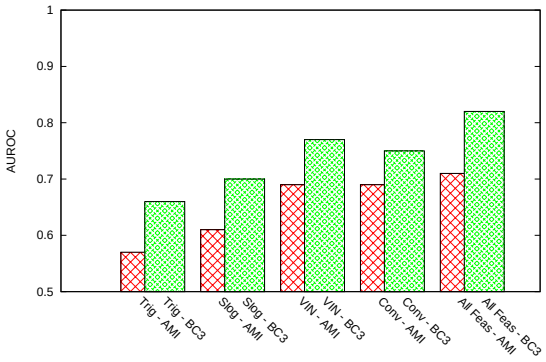


Figure 1: AUROCs on Subjectivity Task for AMI and BC3 corpora

5.1.2 BC3 corpus

For the subjectivity task with the BC3 corpus, the best precision and f-measure scores are found by combining all features, as displayed in Table 5. The f-measure for the VIN approach is ten points higher than for the standard trigram approach.

The average AUROC with the VIN approach is 0.77, compared with 0.70 for AutoSlog-TS (significant at $p<0.05$). The varying instantiation approach is significantly better than the standard trigram pattern approach ($p<0.01$), where the average AUROC is 0.66. We again find that conversational features are very useful for this task, and that the best overall results utilize the entire feature set. These results are displayed in Figure 1.

5.1.3 Impact of Blog Data

An interesting question is whether our use of the BLOG06 data was worthwhile. We can measure this by comparing the VIN AUROC results re-

Sys	Precision	Recall	F-Measure
AMI Corpus			
LB	76	100	86
Trig	87	8	14
Slog	75	46	57
VIN	83	60	70
Conv	82	47	60
All Feas	83	56	67
BC3 Corpus			
LB	54	100	70
Trig	50	84	63
Slog	58	56	57
VIN	53	84	65
Conv	63	80	71
All Feas	60	76	67

Table 6: P/R/F Results, Polarity Task

ported above with the VIN AUROC scores using only the annotated data for learning the significant patterns. The finding is that the blog data was very helpful, as the VIN approach averages only 0.66 on the BC3 data and 0.63 on the AMI data when the blog patterns are *not* used, both significantly lower ($p<0.01$). Figure 2 shows the ROC curves for the VIN approach with and without blog patterns applied to the AMI subjectivity detection task, illustrating the impact of the unsupervised pattern-learning strategy.

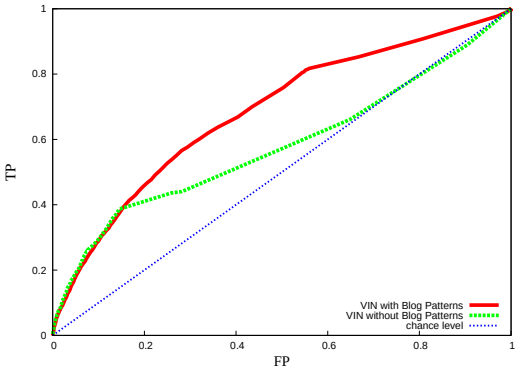


Figure 2: Effect of Blog Patterns on AMI Subjectivity Task

5.2 Positive / Negative Classification

For the polarity classification task, the results differ between the two corpora. We describe the results on meetings and emails in turn.

5.2.1 AMI corpus

The p/r/f results for the AMI polarity task are presented in Table 6, with the scores pertaining to the positive-subjective class. The VIN classifier and full features classifier achieve the highest precision, but the f-measures are below the lower-bound.

Comparing AUROC results, the VIN approach is again significantly better than AutoSlog-TS, averaging 0.65 compared with 0.56, and significantly better than the standard trigram approach ($p<0.01$ in both cases). The results are displayed in Figure 3. The conversational features are significantly less effective than the VIN features ($p<0.05$), and the best overall results are found by utilizing all features, with significant improvement over VIN only at $p<0.05$ and significant improvement over AutoSlog-TS only at $p<0.01$.

5.2.2 BC3 corpus

The results of the polarity task on the BC3 corpus are markedly different from the other experimental results. In this case, neither VIN nor AutoSlog-TS are particularly good for discriminating between positive and negative sentences, and the best strategy is to use features relating to conversational structure. According to p/r/f (Table 6), the only method outperforming the lower-bound in terms of f-measure is the conversational features classifier. According to AUROC scores shown in Figure 3, the conversational features by themselves are significantly better than the VIN approach ($p<0.01$ for non-paired t-test). So for emails, we are more likely to correctly classify positive and negative sentence by looking at features such as position in the turn and participant dominance than by matching our learned patterns. While we showed previously that pattern-based approaches perform well for the subjectivity task on this dataset, there was less success in using the patterns to discern the polarity of email sentences.

We are again interested in whether the use of the BLOG06 data was beneficial. For the BC3 data, there is very little difference between the VIN approach with and without the blog patterns, as they both perform poorly, but with the AMI corpus, the blog patterns yield significant improvement in polarity classification, increasing from an average of 0.56 without the blog patterns to 0.65 with them ($p<0.01$).

6 Discussion and Future Work

A key difference between the AMI and BC3 data with regards to subjectivity is that negative utterances are much more common in the BC3 email threads. Additionally, the pattern-based approaches fared worst in discriminating between

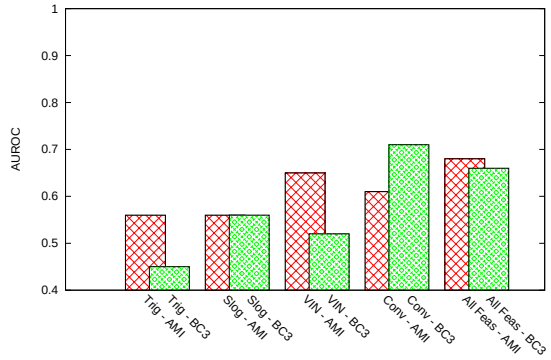


Figure 3: AUROCs on Polarity Task for AMI and BC3 corpora

negative and positive utterances in that corpus. Positive and negative email sentences are more easily recognized via features relating to conversation structure and participant status than through the learned lexical patterns.

The use of patterns learned from unlabeled blog data significantly improved performance. We are currently developing further techniques for learning subjective and polar patterns from such raw, natural text.

A potential area of improvement is to reduce the feature set by eliminating some of the subjective patterns. In Section 2, we briefly described the work of Riloff et al. (2006) on feature subsumption relationships. In our case, in a VIN instantiation set a given trigram instantiation subsumes the less abstract instantiations in the set, so the most abstract instantiation (i.e. completely uninstantiated trigram) representationally subsumes the rest. Eliminating some of the representationally subsumed instantiations when they are also behaviorally subsumed may improve our results.

It is difficult to compare our results directly with those of Raaijmakers et al. (2008) as they used a smaller set of AMI meetings for their experiments, and because for the first experiment we consider the subjective class to be the union of positive-subjective and negative-subjective dialogue acts whereas they additionally include subjective questions and dialogue acts expressing uncertainty. These differences are reflected by the substantially differing scores reported for majority-vote baselines on each task. However, their success with character n-gram features suggests that we could improve our system by incorporating a variety of character features. Character n-grams were the best single feature class in their experiments.

The VIN representation is a general one and may hold promise for learning patterns relevant to other interesting conversation phenomena such as decision-making and action items. We plan to apply the methods described here to these other applications in the near future.

7 Conclusion

In this work we have shown that learning subjective trigrams with varying instantiation levels from both annotated and raw data can improve subjectivity detection and polarity labeling for meeting speech and email threads. The novel pattern-based approach was significantly better than standard trigrams for three of the four tasks, and was significantly better than a state-of-the-art syntactic approach for those same tasks. We also found that features relating to conversational structure were beneficial for all tasks, and particularly for polarity labeling in email data. Interestingly, in three out of four cases combining all the features produced the best performance.

References

F. Biadsy, J. Hirschberg, and E. Filatova. 2008. An unsupervised approach to biography production using wikipedia. In *Proc. of ACL-HLT 2008, Columbus, OH, USA*.

E. Brill. 1992. A simple rule-based part of speech tagger. In *Proc. of DARPA Speech and Natural Language Workshop, San Mateo, CA, USA*, pages 112–116.

J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, I. McCowan, W. Post, D. Reidsma, and P. Wellner. 2005. The AMI meeting corpus: A pre-announcement. In *Proc. of MLMI 2005, Edinburgh, UK*, pages 28–39.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

T. Fawcett. 2003. Roc graphs: Notes and practical considerations for researchers.

G. Murray and G. Carenini. 2008. Summarizing spoken and written conversations. In *Proc. of EMNLP 2008, Honolulu, HI, USA*.

B. Pang and L. Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 1-2(2):1–135.

S. Raaijmakers, K. Truong, and T. Wilson. 2008. Multimodal subjectivity analysis of multiparty conversation. In *Proc. of EMNLP 2008, Honolulu, HI, USA*.

E. Riloff and W. Phillips. 2004. An introduction to the sundance and autoslog systems.

E. Riloff and J. Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proc. of EMNLP 2003, Sapporo, Japan*.

E. Riloff, S. Patwardhan, and J. Wiebe. 2006. Feature subsumption for opinion analysis. In *Proc. of EMNLP 2006, Sydney, Australia*.

S. Somasundaran, J. Ruppenhofer, and J. Wiebe. 2007. Detecting arguing and sentiment in meetings. In *Proc. of SIGDIAL 2007, Antwerp, Belgium*.

J. Ulrich, G. Murray, and G. Carenini. 2008. A publicly available annotated corpus for supervised email summarization. In *Proc. of AAAI EMAIL-2008 Workshop, Chicago, USA*.

T. Wilson, J. Wiebe, and R. Hwa. 2006. Recognizing strong and weak opinion clauses. *Computational Intelligence*, 22(2):73–99.

T. Wilson. 2008. Annotating subjective content in meetings. In *Proc. of LREC 2008, Marrakech, Morocco*.

H. Yu and V. Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proc. of EMNLP 2003, Sapporo, Japan*.

Improved Statistical Machine Translation for Resource-Poor Languages Using Related Resource-Rich Languages

Preslav Nakov
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
nakov@comp.nus.edu.sg

Hwee Tou Ng
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
nght@comp.nus.edu.sg

Abstract

We propose a novel language-independent approach for improving statistical machine translation for resource-poor languages by exploiting their similarity to resource-rich ones. More precisely, we improve the translation from a resource-poor source language X_1 into a resource-rich language Y given a bi-text containing a limited number of parallel sentences for X_1-Y and a larger bi-text for X_2-Y for some resource-rich language X_2 that is closely related to X_1. The evaluation for Indonesian→English (using Malay) and Spanish→English (using Portuguese and pretending Spanish is resource-poor) shows an absolute gain of up to 1.35 and 3.37 Bleu points, respectively, which is an improvement over the rivaling approaches, while using much less additional data.

1 Introduction

Recent developments in statistical machine translation (SMT), *e.g.*, the availability of efficient implementations of integrated open-source toolkits like Moses (Koehn et al., 2007), have made it possible to build a prototype system with decent translation quality for any language pair in a few days or even hours. In theory. In practice, doing so requires having a large set of parallel sentence-aligned bi-lingual texts (a *bi-text*) for that language pair, which is often unavailable. Large high-quality bi-texts are rare; except for Arabic, Chinese, and some official languages of the European Union (EU), most of the 6,500+ world languages remain resource-poor from an SMT viewpoint.

While manually creating a small bi-text could be relatively easy, building a *large* one is hard, *e.g.*, because of copyright. Most bi-texts for SMT come from parliament debates and legislation of multi-lingual countries (*e.g.*, French-English from Canada, and Chinese-English from Hong Kong), or from international organizations like the United Nations and the European Union. For example, the *Europarl* corpus of parliament proceedings consists of about 1.3M parallel sentences (up to 44M words) per language for 11 languages (Koehn, 2005), and the *JRC-Acquis* corpus provides a comparable amount of European legislation in 22 languages (Steinberger et al., 2006).

The official languages of the EU are especially lucky in that respect; while this includes such "classic SMT languages" like English and French, and some important international ones like Spanish and Portuguese, most of the rest have a limited number of speakers and were resource-poor until recently; this is changing quickly because of the increasing volume of EU parliament debates and the ever-growing European legislation. Thus, becoming an official language of the EU has turned out to be an easy recipe for getting resource-rich in bi-texts quickly. Of course, not all languages are that 'lucky', but many can still benefit.

In this paper, we propose using bi-texts for resource-rich language pairs to build better SMT systems for resource-poor ones by exploiting the similarity between a resource-poor language and a resource-rich one.

The proposed method allows non-EU languages to benefit from being closely related to one or more official languages of the EU, the most obvious candidates being Norwegian (related to Swedish), Moldavian[1] (related to Romanian), and Macedonian[2] (related to Bulgarian). After Croatia joins the EU, Serbian, Bosnian and Montenegrin will be able to benefit from Croatian gradually turning resource-rich (all four split from Serbo-Croatian after the breakup of Yugoslavia). The newly-made EU-official (and thus not as resource-

[1] Not recognized by Romania.
[2] Not recognized by Bulgaria and Greece.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1358–1367,
Singapore, 6-7 August 2009.

rich) Czech and Slovak are another possible pair of candidates. As we will see below, even such resource-rich languages like Spanish and Portuguese can benefit from the proposed method. Of course, many pairs of closely related languages can be also found outside of Europe, Malay and Indonesian being just one such example we will experiment with.

The remainder of the present paper is organized as follows: Section 2 presents our method, Section 3 describes the experiments, and Section 4 discusses the results and the general applicability of the approach. Section 5 provides an overview of the related work. Finally, Section 6 concludes and suggests possible directions for future work.

2 Method

We propose a novel language-independent approach for improving statistical machine translation for resource-poor languages by exploiting their similarity to resource-rich ones. More precisely, we improve the translation from a resource-poor source language X_1 into a resource-rich target language Y given a bi-text containing a limited number of parallel sentences for X_1-Y and a much larger bi-text for X_2-Y for some resource-rich language X_2 that is closely related to X_1.

Our method exploits the similarity between related languages with respect to word order, syntax, and, most importantly, vocabulary overlap – related languages share a large number of cognates.

Before we present the method, we will describe two simple strategies for integrating the bi-text for X_2-Y into a phrase-based SMT system for X_1-Y.

2.1 Merging Bi-texts

We can simply concatenate the bi-texts for X_1-Y and X_2-Y into one large bi-text and use it to train an SMT system.

This offers several advantages. First, it can yield improved word alignments for the sentences that came from the X_1-Y bi-text, *e.g.*, since the additional sentences can provide new contexts for the rare words in that bi-text; rare words are hard to align, which could have a disastrous effect on the subsequent phrase extraction stage. Second, it can provide new source-language side translation options, thus increasing the lexical coverage and reducing the number of unknown words at translation time; it can also provide new useful non-compositional phrases on the source-language side, thus yielding more fluent translation output. Third, it can offer new target-language side phrases for known source phrases, which could improve fluency by providing more translation options for the language model (LM) to choose from. Fourth, bad phrases including words from X_2 that do not exist in X_1 will be effectively ignored at translation time since they could never possibly match the input, while bad new target-language translations still have the chance to be filtered out by the language model.

However, simple concatenation can be problematic. First, when concatenating the small bi-text for X_1-Y with the much larger one for X_2-Y, the latter will dominate during word alignment and phrase extraction, thus hugely influencing both lexical and phrase translation probabilities, which can yield poor performance. This can be counteracted by repeating the small bi-text several times so that the large one does not dominate. Second, since the bi-texts are merged mechanically, there is no way to distinguish between phrases extracted from the bi-text for X_1-Y (which should be good), from those coming from the bi-text for X_2-Y (whose quality might be questionable).

2.2 Combining Phrase Tables

An alternative way of making use of the additional bi-text for X_2-Y to train an improved SMT system for $X_1 \rightarrow Y$ is to build separate phrase tables from X_1-Y and X_2-Y, which can then be (a) used together, *e.g.*, as alternative decoding paths, (b) merged, *e.g.*, using one or more extra features to indicate the bi-text each phrase came from, or (c) interpolated, *e.g.*, using simple linear interpolation.

Building two separate phrase tables offers several advantages. First, the good phrases from the bi-text for X_1-Y are clearly distinguished from (or given a higher weight in the linear interpolation compared to) the potentially bad ones from the X_2-Y bi-text. Second, the lexical and the phrase translation probabilities are combined in a principled manner. Third, using an X_2-Y bi-text that is much larger than that for X_1-Y is not problematic any more. Fourth, as with bi-text merging, there are many additional source- and target-language phrases, which offer new translation options.

On the negative side, the opportunity is lost to obtain improved word alignments for the sentences in the X_1-Y bi-text.

2.3 Proposed Method

Taking into account the potential advantages and disadvantages of the above strategies, we propose a method that tries to get the best of both: (i) increased lexical coverage by using additional phrase pairs independently extracted from X_2-Y, and (ii) improved word alignments for X_1-Y by biasing the word alignment process with additional sentence pairs from X_2-Y (possibly also repeating X_1-Y several times). A detailed description of the method follows:

1. Build a bi-text B_{cat} that is a concatenation of the bi-texts for X_1-Y and X_2-Y. Generate word alignments for B_{cat}, extract phrases, and build a phrase table T_{cat}.

2. Build a bi-text B_{rep} from the X_1-Y bi-text repeated k times followed by one copy of the X_2-Y bi-text. Generate word alignments for B_{rep}, then truncate them, only keeping word alignments for one copy of the X_1-Y bi-text. Use these word alignments to extract phrases, and build a phrase table T_{rep_trunc}.

3. Produce a phrase table T_{merged} by combining T_{cat} and T_{rep_trunc}, giving priority to the latter, and use it in an $X_1 \rightarrow Y$ SMT system.

2.4 Transliteration

As we mentioned above, our method relies on the existence of a large number of *cognates* between related languages. While linguists define cognates as words derived from a common root[3] (Bickford and Tuggy, 2002), *computational* linguists typically ignore origin, defining them as words in different languages that are mutual translations and have a similar orthography (Bergsma and Kondrak, 2007; Mann and Yarowsky, 2001; Melamed, 1999). In this paper, we adopt the latter definition.

Cognates between related languages often exhibit minor spelling variations, which can be simply due to different rules of orthography, (*e.g.*, *senhor vs. señor* in Portuguese and Spanish), but often stem from real phonological differences. For example, the Portuguese suffix *-ção* corresponds to the Spanish suffix *-ción*, *e.g.*, *evolução* vs. *evolución*. Such correspondences can be quite frequent and thus easy to learn automatically[4]. Even more frequent can be the inflectional variations. For example, in Portuguese and Spanish respectively, verb endings like *-ou* vs. *-ó* (for 3rd person singular, simple past tense), *e.g.*, *visitou* vs. *visitó*, or *-ei* vs. *-é* (for 1st person singular, simple past tense), *e.g.*, *visitei* vs. *visité*.

If such systematic differences exist between the languages X_1 and X_2, it might be useful to learn and to use them as a pre-processing step in order to transliterate the X_2 side of the X_2-Y bi-text and thus increase its vocabulary overlap with the source language X_1.

We will describe our approach to automatic transliteration in more detail in Section 3.4 below.

3 Experiments

3.1 Language Pairs

We experimented with two language pairs: the closely related Malay and Indonesian and the more dissimilar Spanish and Portuguese.

Malay and Indonesian are mutually intelligible, but differ in pronunciation and vocabulary. An example follows[5]:

- **Malay:** *Semua manusia dilahirkan bebas dan samarata dari segi kemuliaan dan hak-hak.*

- **Indonesian:** *Semua orang dilahirkan merdeka dan mempunyai martabat dan hak-hak yang sama.*

Spanish and Portuguese also exhibit a noticeable degree of mutual intelligibility, but differ in pronunciation, spelling, and vocabulary. Unlike Malay and Indonesian, however, they also differ syntactically and have a high degree of spelling differences as demonstrated by the following examples[6]:

- **Spanish:** *Señora Presidenta, estimados colegas, lo que está sucediendo en Oriente Medio es una tragedia.*

- **Portuguese:** *Senhora Presidente, caros colegas, o que está a acontecer no Medio Oriente é uma tragédia.*

[3] E.g., Latin *tu*, Old English *thou*, Spanish *tú*, Greek *sú* and German *du* are all cognates meaning '2^{nd} person singular'.

[4] Not all such differences are systematic; many apply to a particular word only, *e.g.*, *kerana vs. kareña* in Malay and Indonesian, or *dizer vs. decir* in Portuguese and Spanish.

[5] In English: *All human beings are born free and equal in dignity and rights.* (from Article 1 of the Universal Declaration of Human Rights)

[6] In English: *Madam President, ladies and gentlemen, the events in the Middle East are a real tragedy.*

3.2 Datasets

In our experiments, we used the following number of training sentence pairs (number of words shown in parentheses) for English (en), Indonesian (in), Malay (ml), Portuguese(pt), and Spanish (es):

- **Indonesian-English (*in-en*):**
  - 28,383 pairs (0.8M, 0.9M words);
  - monolingual English en_{in}: 5.1M words.
- **Malay-English (*ml-en*):**
  - 190,503 pairs (5.4M, 5.8M words);
  - monoling. English en_{ml}: 27.9M words.
- **Spanish-English (*es-en*):**
  - 1,240,518 pairs (35.7M, 34.6M words);
  - monolingual English $en_{es:pt}$: 45.3M words (the same as for *pt-en*).
- **Portuguese-English (*pt-en*):**
  - 1,230,038 pairs (35.9M, 34.6M words).
  - monolingual English $en_{es:pt}$: 45.3M words (the same as for *es-en*).

All of the above datasets contain sentences with up to 100 tokens. In addition, for each of the four language pairs, we have a development and a testing bi-text, each with 2,000 parallel sentence pairs. We made sure the development and the testing bi-texts shared no sentences with the training bi-texts; we further excluded from the monolingual English data all sentences from the English sides of the training and the development bi-texts.

The training bi-text datasets for *es-en* and *pt-en* were built from release v.3 of the *Europarl* corpus, excluding the Q4/2000 portion out of which we created our testing and development datasets.

We built the *in-en* bi-texts from texts that we downloaded from the Web. We translated the Indonesian texts to English using *Google Translate*, and we matched[7] them against the English texts using a cosine similarity measure and heuristic constraints based on document length in words and in sentences, overlap of numbers, words in uppercase, and words in the title. Next, we extracted pairs of sentences from the matched document pairs using *competitive linking* (Melamed, 2000), and we retained the ones whose similarity was above a pre-specified threshold. The *ml-en* was built in a similar manner.

[7]Note that the automatic translations were used for matching only; the final bi-text contained no automatic translations.

3.3 Baseline SMT System

In the baseline, we used the following setup: We first tokenized and lowercased both sides of the training bi-text. We then built separate directed word alignments for English$\rightarrow X$ and $X\rightarrow$English ($X\in\{$Indonesian, Spanish$\}$) using IBM model 4 (Brown et al., 1993), combined them using the *intersect+grow heuristic* (Och and Ney, 2003), and extracted phrase-level translation pairs of maximum length seven using the *alignment template approach* (Och and Ney, 2004). We thus obtained a phrase table where each pair is associated with five parameters: forward and reverse phrase translation probabilities, forward and reverse lexical translation probabilities, and phrase penalty.

We then trained a log-linear model using standard SMT feature functions: trigram language model probability, word penalty, distance-based[8] distortion cost, and the parameters from the phrase table. We set all weights by optimizing Bleu (Papineni et al., 2002) using minimum error rate training (MERT) (Och, 2003) on a separate development set of 2,000 sentences (Indonesian or Spanish), and we used them in a beam search decoder (Koehn et al., 2007) to translate 2,000 test sentences (Indonesian or Spanish) into English. Finally, we detokenized the output, and we evaluated it against a lowercased gold standard using Bleu[9].

3.4 Transliteration

As was mentioned in Section 2, transliteration can be helpful for languages with regular spelling differences. Thus, we built a system for transliteration from Portuguese into Spanish that was trained on a list of automatically extracted likely cognates. The system was applied on the Portuguese side of the *pt-en* training bi-text.

Classic approaches to automatic cognate extraction look for non-stopwords with similar spelling that appear in parallel sentences in a bi-text (Kondrak et al., 2003). In our case, however, we need to extract cognates between Spanish and Portuguese given *pt-en* and *es-en* bi-texts only, *i.e.*, without having a *pt-es* bi-text. Although it is easy to construct a *pt-es* bi-text from the Europarl corpus, we chose not to do so since, in general, synthe-

[8]We also tried lexicalized reordering (Koehn et al., 2005). While it yielded higher absolute Bleu scores, the relative improvement for a sample of our experiments was very similar to that achieved with distance-based re-ordering.

[9]We used version 11b of the NIST scoring tool: `http://www.nist.gov/speech/tools/`

sizing a bi-text for X_1-X_2 would be impossible: *e.g.*, it cannot be done for *ml-in* given our training datasets for *in-en* and *ml-en* since the English sides of these bi-texts have no sentences in common.

Thus, we extracted the list of likely cognates between Portuguese and Spanish from the training *pt-en* and *es-en* bi-texts using English as a pivot as follows: We started with IBM model 4 word alignments, from which we extracted four conditional lexical translation probabilities: $\Pr(p_j|e_i)$ and $\Pr(e_i|p_j)$ for Portuguese-English, and $\Pr(s_k|e_i)$ and $\Pr(e_i|s_k)$ for Spanish-English, where p_j, e_i and s_k stand for a Portuguese, an English and a Spanish word respectively. Following Wu and Wang (2007), we then induced conditional lexical translation probabilities $\Pr(p_j|s_k)$ and $\Pr(s_k|p_j)$ for Portuguese-Spanish as follows:

$$\Pr(p_j|s_k) = \sum_i \Pr(p_j|e_i, s_k)\Pr(e_i|s_k)$$

Assuming p_j is conditionally independent of s_k given e_i, we can simplify the above expression:

$$\Pr(p_j|s_k) = \sum_i \Pr(p_j|e_i)\Pr(e_i|s_k)$$

Similarly, for $\Pr(s_k|p_j)$, we obtain

$$\Pr(s_k|p_j) = \sum_i \Pr(s_k|e_i)\Pr(e_i|p_j)$$

We excluded all stopwords, words of length less than three, and those containing digits. We further calculated $\mathrm{Prod}(p_j, s_k) = \Pr(p_j|s_k)\Pr(s_k|p_j)$, and we excluded all Portuguese-Spanish word pairs (p_j, s_k) for which $\mathrm{Prod}(p_j, s_k) < 0.01$. From the remaining pairs, we extracted likely cognates based on $\mathrm{Prod}(p_j, s_k)$ and on the orthographic similarity between p_j and s_k.

Following Melamed (1995), we measured the orthographic similarity using the *longest common subsequence ratio* (LCSR), defined as follows:

$$\mathrm{LCSR}(s_1, s_2) = \frac{|\mathrm{LCS}(s_1,s_2)|}{\max(|s_1|,|s_2|)}$$

where $\mathrm{LCS}(s_1, s_2)$ is the *longest common subsequence* of s_1 and s_2, and $|s|$ is the length of s.

We retained as likely cognates all pairs for which LCSR was 0.58 or higher; that value was found by Kondrak et al. (2003) to be optimal for a number of language pairs in the *Europarl* corpus.

Finally, we performed *competitive linking* (Melamed, 2000), assuming that each Portuguese wordform had at most one Spanish best cognate match. Thus, using the values of $\mathrm{Prod}(p_j, s_k)$, we induced a fully-connected weighted bipartite graph. Then, we performed a greedy approximation to the maximum weighted bipartite matching in that graph (*i.e.*, competitive linking) as follows: First, we accepted as cognates the cross-lingual pair (p_j, s_k) with the highest $\mathrm{Prod}(p_j, s_k)$ in the graph, and we discarded p_j and s_k from further consideration. Then, we accepted the next highest-scored pair, and we discarded the involved wordforms and so forth. The process was repeated until there were no matchable pairs left.

As a result of the above procedure, we ended up with 28,725 Portuguese-Spanish cognate pairs, 9,201 (or 32%) of which had spelling differences. For each pair in the list of cognate pairs, we added spaces between any two adjacent letters for both wordforms, and we further appended the start and the end characters ^ and $. For example, the cognate pair *evolução – evolución* became

^ e v o l u ç ã o $ — ^ e v o l u c i ó n $

We randomly split the resulting list into a training (26,725 pairs) and a development dataset (2,000 pairs), and trained and tuned a character-level phrase-based monotone SMT system similar to (Finch and Sumita, 2008) to transliterate a Portuguese wordform into a Spanish wordform. We used a Spanish language model trained on 14M word tokens (obtained from the above-mentioned 45.3M-token monolingual English corpus after excluding punctuation, stopwords, words of length less than three, and those containing digits): one per line and character-separated with added start and end characters as in the above example. We set both the maximum phrase length and the language model order to ten; this value was found by tuning on the development dataset. The system was tuned using MERT, and the feature weights were saved. The tuning Bleu was 95.22%, while the baseline Bleu, for leaving the Portuguese words intact, was 87.63%. Finally, the training and the tuning datasets were merged, and a new training round was performed. The resulting system was used with the saved feature weights to transliterate the Portuguese side of the training *pt-en* bi-text, which yielded a new pt_{es}-*en* training bi-text.

We did the same for Malay into Indonesian. We extracted 5,847 cognate pairs, 844 (or 14.4%) of which had spelling differences, and we trained a transliteration system. The highest tuning Bleu was 95.18% (for maximum phrase size and LM order of 10), but the baseline was 93.15%. We then re-trained the system on the combination of the training and the development datasets, and we transliterated the Malay side of the training *ml-en* bi-text, obtaining a new ml_{in}-*en* training bi-text.

#	Train	LM	Dev	Test	10K	20K	40K	80K	160K	320K	640K	1230K
1	ml-en	en_{ml}	ml-en	ml-en	44.93	46.98	47.15	48.04	49.01	–	–	–
2	ml_{in}-en	en_{ml}	ml-en	ml-en	38.99	40.96	41.02	41.88	42.81	–	–	–
3	ml-en	en_{ml}	ml-en	**in-en**	13.69	14.58	14.76	15.12	15.84	–	–	–
4	ml-en	en_{ml}	**in-en**	**in-en**	13.98	14.75	14.91	15.51	16.27	–	–	–
5	ml-en	**en**$_{in}$	**in-en**	**in-en**	15.56	16.38	16.52	17.04	17.90	–	–	–
6	ml_{in}-en	**en**$_{in}$	**in-en**	**in-en**	16.44	17.36	17.62	18.14	19.15	–	–	–
7	pt-en	$\text{en}_{es:pt}$	pt-en	pt-en	21.28	23.11	24.43	25.72	26.43	27.10	27.78	27.96
8	pt_{es}-en	$\text{en}_{es:pt}$	pt-en	pt-en	10.91	11.56	12.16	12.50	12.83	13.27	13.48	13.71
9	pt-en	**en**$_{es:pt}$	pt-en	**es-en**	4.40	4.77	4.57	5.02	4.99	5.32	5.08	5.34
10	pt-en	**en**$_{es:pt}$	**es-en**	**es-en**	4.91	5.12	5.64	5.82	6.35	6.87	6.44	7.10
11	pt_{es}-en	**en**$_{es:pt}$	**es-en**	**es-en**	8.18	9.03	9.97	10.66	11.35	12.26	12.69	13.79

Table 1: **Cross-lingual SMT experiments (shown in bold).** Columns 2-5 present the bi-texts used for training, development and testing, and the monolingual data used to train the English language model. The following columns show the resulting Bleu (in %s) for different numbers of training sentence pairs.

3.5 Cross-lingual Translation

In this subsection, we study the similarity between the original and the additional source languages.

First, we measured the vocabulary overlap between Spanish and Portuguese, which was feasible since our training *pt-en* and *es-en* bi-texts are from the same time span in the *Europarl* corpus and their English sides largely overlap. We found 110,053 Portuguese and 121,444 Spanish word types, and 44,461 (or 36.6%) of them were identical. Unfortunately, we could not do the same for Malay and Indonesian since the English sides of the *in-en* and *ml-en* bi-texts do not overlap.

Second, following the setup of the baseline system, we performed cross-lingual experiments. The results are shown in Table 1. As we can see, this yielded a huge decrease in Bleu compared to the baseline – three to five times – even for very large training datasets, and even when a proper English LM and development dataset were used: compare line 1 to lines 3-6, and line 7 to lines 9-11.

Third, we tried transliteration. Bleu doubled for Spanish (see lines 10-11), but improved far less for Indonesian (lines 5-6). Training on the transliterated data and testing on Malay and Portuguese yielded about 10-12% relative decrease for Malay (lines 1-2) but 50% for Portuguese (lines 7-8).[10] Thus, unlike Spanish and Portuguese, there were far less systematic spelling variations between Malay and Indonesian. A closer inspection confirmed this: many extracted likely Malay-Indonesian cognate pairs with spelling differences were in fact forms of a word existing in both languages, *e.g.*, *kata* and *berkata* ('*to say*').

[10]However, as lines 8 and 11 show, a system trained on 1.23M pt_{es}*-en* sentence pairs, performs equally well when translating Portuguese and Spanish text: 13.71% *vs.* 13.79%.

3.6 Using an Additional Language

We performed various experiments combining the original and an additional training bi-text:

Two-tables: We built two separate phrase tables for the two bi-texts, and we used them in the alternative decoding path model of Birch et al. (2007).

Interpolation: We built two separate phrase tables for the original and for the additional bi-text, and we used linear interpolation to combine the corresponding conditional probabilities: $\Pr(e|s) = \alpha \Pr_{orig}(e|s) + (1-\alpha)\Pr_{extra}(e|s)$. We optimized the value of α on the development dataset, trying .5, .6, .7, .8 and .9; we used the same α for all four conditional probabilities.

Merge: We built separate phrase tables, T_{orig} and T_{extra}, for the original and for the additional training bi-text. We then concatenated them giving priority to T_{orig}: We kept all phrase pairs from T_{orig}, adding to them those ones from T_{extra} that were not present in T_{orig}. For each phrase pair added, we retained its associated conditional probabilities and the phrase penalty. We further added three additional features to each entry in the new table: F_1, F_2 and F_3. The value of F_1 was 1 if the phrase pair came from T_{orig}, and 0.5 otherwise. Similarly, F_2=1 if the phrase pair came from T_{extra}, and F_2=0.5 otherwise. The value of F_3 was 1 if the pair came from both T_{orig} and T_{extra}, and 0.5 otherwise. We experimented using (1) F_1 only, (2) F_1 and F_2, (3) F_1, F_2, and F_3. We set all feature weights using MERT, and we optimized the number of features on the development set.[11]

[11]In theory, we should have also re-normalized the probabilities since they may not sum to one. In practice, this was not that important since the log-linear SMT model does not require that the features be probabilities at all (*e.g.*, the phrase penalty), and we had extra features whose impact was bigger.

Concat$\times k$: We concatenated k copies of the original and one copy of the additional training bi-text; we then trained and tuned an SMT system as for the baseline. The value for k was optimized on the development dataset.

Concat$\times k$**:align**: We concatenated k copies of the original and one copy of the additional training bi-text. We then generated IBM model 4 word alignments, and we truncated them, only keeping them for one copy of the original training bi-text. Next, we extracted phrase pairs, thus buildng a phrase table, and we tuned an SMT system as for the baseline.

Our Method: Our method was described in Section 2. We used **merge** to combine the phrase tables for **concat**$\times k$**:align** and **concat**$\times$**1**, considering the former as T_{orig} and the latter as T_{extra}. We had two parameters to tune: k and the number of extra features in the merged phrase table.

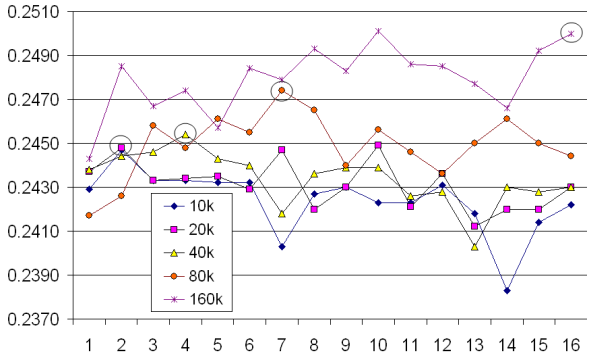


Figure 1: **Impact of k on Bleu for *concat*$\times k$ for different number of extra *ml-en* sentence pairs in Indonesian→English SMT.**

4 Results and Discussion

First, we studied the impact of k on **concat**$\times k$ for Indonesian→English SMT using Malay as an additional language. We tried all values of k such that $1 \leq k \leq 16$ with $10000n$ extra *ml-en* sentence pairs, $n \in \{1,2,4,8,16\}$. As we can see in Figure 1, the highest Bleu scores are achieved for $(n;k) \in \{(1;2),(2;2),(4;4),(8;7),(16;16)\}$, *i.e.*, when $k \approx n$. In order to limit the search space, we used this relationship between k and n in our experiments (also for Portuguese and Spanish).

Table 2 shows the results for experiments on improving Indonesian→English SMT using 10K, 20K, ..., 160K additional *ml-en* pairs of parallel sentences. Several observations can be made. First, using more additional sentences yields better results. Second, with one exception, all experiments yield improvements over the baseline. Third, the improvements are always statistically significant for **our method**, according to (Collins et al., 2005)'s sign test. Overall, among the different bi-text combination strategies, **our method** performs best, followed by **concat**$\times k$, **merge**, and **interpolate**, which are very close in performance; these three strategies are the only ones to consistently yield higher Bleu as the number of additional *ml-en* sentence pairs grows. Methods like **concat**$\times$1, **concat**$\times k$**:align** and **two-tables** are somewhat inconsistent in that respect. The latter method performs worst and is the only one to go below the baseline (for 10K *ml-en* pairs).

Table 3 shows the results when using *pt-en* data to improve Spanish→English SMT. Overall, the results and the conclusions that can be made are consistent with those for Table 2. We can further observe that, as the size of the original bi-text increases, the gain in Bleu decreases, which is to be expected. Note also that here transliteration is very important: it doubles the absolute gain in Bleu.

Finally, Table 4 shows a comparison to the pivoting technique of Callison-Burch et al. (2006). for English→Spanish SMT. Despite using just Portuguese, we achieve an improvement that is, in five out of six cases, much better than what they achieve with *eight* pivot languages (which include not only Portuguese, but also two other Romance languages, French and Italian, which are closely related to Spanish). Moreover, our method yields improvements for very large original datasets – 1.2M pairs, while theirs stops improving at 160K. However, our improvements are only statistically significant for 160K original pairs or less. Finally, note that our translation direction is reversed.

Based on the experimental results, we can make several conclusions. First, we have shown that using bi-text data from related languages improves SMT: we achieved up to 1.35 and 3.37 improvement in Bleu for *in-en* (+*ml-en*) and *es-en* (+*pt-en*) respectively. Second, while simple concatenation can help, it is problematic when the additional sentences out-number the ones from the original bi-text. Third, concatenation can work very well if the original bi-text is repeated enough times so that the additional bi-text does not dominate. Fourth, merging phrase tables giving priority to the original bi-text and using additional fea-

in-en	*ml-en*	Baseline	Two tables	Interpol.	Merge	concat×1	concat×k	concat×k:align	Our method
28.4K	10K	23.80$^{<}$	$^{\leq}$23.79$^{<}$	23.89$^{<}_{(.9)}$	23.97$^{<}_{(3)}$	24.29$^{<}$	24.29$^{<}_{(1)}$	24.01$^{<}_{(1)}$	$^{<}$**24.51**$_{(2;1)}$ (+0.72)
28.4K	20K	23.80$^{<}$	24.24$^{<}$	24.22$^{<}_{(.8)}$	$^{\leq}$24.46$^{<}_{(3)}$	24.37$^{<}$	$^{\leq}$24.48$_{(2)}$	$^{<}$24.35$^{<}_{(2)}$	$^{<}$**24.70**$_{(2;2)}$ (+0.90)
28.4K	40K	23.80$^{<}$	24.27$^{<}$	24.27$^{<}_{(.8)}$	24.43$^{\leq}_{(3)}$	24.38$^{\leq}$	$^{\leq}$24.54$_{(4)}$	$^{<}$24.39$^{<}_{(4)}$	$^{<}$**24.73**$_{(4;2)}$ (+0.93)
28.4K	80K	23.80$^{<}$	24.11$^{<}$	$^{\leq}$24.46$^{<}_{(.8)}$	$^{<}$24.67$_{(3)}$	24.17$^{<}$	$^{\leq}$24.65$^{<}_{(8)}$	24.18$^{<}_{(8)}$	$^{<}$**24.97**$_{(8;3)}$ (+1.17)
28.4K	160K	23.80$^{<}$	$^{<}$24.58$^{<}$	$^{<}$24.58$^{<}_{(.8)}$	$^{<}$24.79$^{\leq}_{(3)}$	$^{\leq}$24.43$^{<}$	$^{<}$25.00$_{(16)}$	$^{\leq}$24.27$^{<}_{(16)}$	$^{<}$**25.15**$_{(16;3)}$ (+1.35)

Table 2: **Improving Indonesian→English SMT using *ml-en* data.** Shown are the Bleu scores (in %s) for different methods. A subscript shows the best parameter value(s) found on the development set and used on the test set to produce the given result. Bleu scores that are statistically significantly better than the baseline/our method are marked on the left/right side by $^{<}$ (for $p < 0.01$) or $^{\leq}$ (for $p < 0.05$).

es-en	*pt-en*	Transl.	Baseline	Two tables	Interpol.	Merge	concat×1	concat×k	concat×k:align	Our method
10K	160K	no	22.87$^{<}$	$^{<}$23.81	$^{<}$23.73$_{(.5)}$	$^{<}$23.60$_{(2)}$	$^{<}$23.54$^{<}$	$^{<}$23.83$^{<}_{(16)}$	22.93$^{<}_{(16)}$	$^{<}$**23.98**$_{(16;3)}$ (+1.11)
		yes	22.87$^{<}$	$^{<}$25.29$^{\leq}$	$^{<}$25.22$^{<}_{(.5)}$	$^{<}$25.16$^{<}_{(2)}$	$^{<}$25.26	$^{<}$25.42$_{(16)}$	$^{<}$23.31$^{<}_{(16)}$	$^{<}$**25.73**$_{(16;3)}$ (+2.86)
20K	160K	no	24.71$^{<}$	$^{<}$25.22	$^{\leq}$25.02$^{<}_{(.5)}$	$^{<}$25.32$^{\leq}_{(3)}$	$^{<}$25.19$^{<}$	$^{<}$25.29$^{<}_{(8)}$	24.91$^{<}_{(8)}$	$^{<}$**25.65**$_{(8;2)}$ (+0.94)
		yes	24.71$^{<}$	$^{<}$26.07$^{\leq}$	$^{<}$26.07$_{(.7)}$	$^{<}$26.04$^{<}_{(3)}$	$^{<}$26.16$^{\leq}$	$^{<}$26.18$^{\leq}_{(8)}$	24.88$^{<}_{(8)}$	$^{<}$**26.36**$_{(8;3)}$ (+1.65)
40K	160K	no	25.80$^{<}$	25.96$^{<}$	26.15$^{<}_{(.6)}$	25.99$^{<}_{(3)}$	26.24$^{<}$	25.92$^{<}_{(4)}$	25.99$^{<}_{(4)}$	$^{<}$**26.49**$_{(4;2)}$ (+0.69)
		yes	25.80$^{<}$	$^{<}$26.68	$^{<}$26.43$_{(.7)}$	$^{<}$26.64$_{(3)}$	$^{<}$26.78	$^{<}$26.93$_{(4)}$	25.88$^{<}_{(4)}$	$^{<}$**26.95**$_{(4;3)}$ (+1.15)
80K	160K	no	27.08$^{\leq}$	$^{\geq}$26.89$^{<}$	27.04$^{<}_{(.8)}$	27.02$^{<}_{(3)}$	27.23	27.09$^{<}_{(2)}$	27.01$^{<}_{(2)}$	$^{\leq}$**27.30**$_{(2;2)}$ (+0.22)
		yes	27.08$^{<}$	27.20$^{<}$	27.42$_{(.5)}$	27.29$^{\leq}_{(3)}$	27.26$^{<}$	$^{\leq}$27.53$_{(2)}$	27.09$^{<}_{(2)}$	$^{<}$**27.49**$_{(2;3)}$ (+0.41)
160K	160K	no	27.90	27.99	27.72$_{(.5)}$	27.95$_{(2)}$	27.83$^{<}$	27.83$^{<}_{(1)}$	27.94$_{(1)}$	**28.05**$_{(1;3)}$ (+0.15)
		yes	27.90	28.11	$^{\leq}$28.13$_{(.6)}$	$^{\leq}$28.17$_{(2)}$	$^{\leq}$28.14	$^{\leq}$28.14$_{(1)}$	28.06$_{(1)}$	**28.16**$_{(1;2)}$ (+0.26)

Table 3: **Improving Spanish→English SMT using 160K additional *pt-en* sentence pairs.** Column three shows whether transliteration was used; the following columns list the Bleu scores (in %s) for different methods. A small subscript shows the best parameter value(s) found on the development set and used on the test set to produce the given result. Bleu scores that are statistically significantly better than the baseline/our method are marked on the left/right side by $^{<}$ (for $p < 0.01$) or $^{\leq}$ (for $p < 0.05$).

tures is a good strategy. Fifth, part of the improvement when combining bi-texts is due to increased vocabulary coverage because of cognates, but another part comes from improved word alignments. Sixth, the best results are achieved when the latter two sources are first isolated and then combined (our method). Finally, transliteration can help a lot in case of systematic spelling variations between the original and the additional source languages.

5 Related Work

In this section, we describe two general lines of related previous research: using cognates between the source and the target language, and source-language side paraphrasing with a pivot language.

5.1 Cognates

Many researchers have used likely cognates to obtain improved word alignments and thus build better SMT systems. Al-Onaizan et al. (1999) extracted such likely cognates for Czech-English using one of the variations of LCSR (Melamed, 1995) described in (Tiedemann, 1999) as a similarity measure. They used these cognates to improve word alignments with IBM models 1-4 in three different ways: (1) by seeding the parameters of IBM model 1, (2) by constraining the word co-occurrences when training IBM models 1-4, and (3) by adding the cognate pairs to the bi-text as additional "sentence pairs". The last approach performed best and was later used by Kondrak et al. (2003) who demonstrated improved SMT for nine European languages.

Unlike these approaches, which extract cognates between the source and the target language, we use cognates between the source and some other related language that is different from the target. Moreover, we only implicitly rely on the existence of such cognates; we do not try to extract them at all, and we leave them in their original sentence contexts.[12]

[12]However, in some of our experiments, we extract cognates for training a transliteration system from the resource-rich source language X_2 into the resource-poor one X_1.

Direction	System	10K	20K	40K	80K	160K	320K	1,230K
en→es	*baseline*	22.6	25.0	26.5	26.5	28.7	30.0	–
	pivoting (+8 languages × ∼1.3M pairs)	23.3	26.0	27.2	28.0	29.0	30.0	–
	improvement	**+0.7**	**+1.0**	**+0.7**	**+1.5**	**+0.3**	**+0.0**	–
es→en	baseline	22.87	24.71	25.80	27.08	27.90	28.46	29.90
	our method (+1 language × **160K** pairs)	23.98*	25.65*	26.49*	27.30$^\diamond$	28.05	28.52	29.87
	improvement	**+1.11***	**+0.94***	**+0.69***	**+0.22**$^\diamond$	**+0.15**	**+0.06**	**-0.03**
	our method (**translit.**, +1 lang. × **160K**)	25.73*	26.36*	26.95*	27.49*	28.16	28.43	29.94
	improvement	**+2.86***	**+1.65***	**+1.15***	**+0.41***	**+0.26**	**-0.03**	**+0.04**
	our method (+1 language × **1.23M** pairs)	24.23*	25.70*	26.78*	27.49	28.22$^\diamond$	28.58	29.84
	improvement	**+1.36***	**+0.99***	**+0.98***	**+0.41**	**+0.32**$^\diamond$	**+0.12**	**-0.06**
	our method (**translit.**, +1 lang. × **1.23M**)	26.24*	26.82*	27.47*	27.85*	28.50*	28.70	29.99
	improvement	**+3.37***	**+2.11***	**+1.67***	**+0.77***	**+0.60***	**+0.24**	**+0.09**

Table 4: **Comparison to the pivoting technique of Callison-Burch et al. (2006) for English→Spanish.** Shown are Bleu scores (in %s) and absolute improvement over the baseline for training bi-texts with different numbers of parallel sentences (10K, 20K, . . ., 1230K) and fixed amount of additional data: (1) about 1.3M sentence pairs for each of eight additional languages in Callison-Burch et al. (2006)'s pivoting, and (2) 160K and 1,230K pairs for one language (Portuguese) for our method. Statistically significant improvements over the baseline are marked with a * (for $p < 0.01$) and with a $^\diamond$ (for $p < 0.05$).

5.2 Paraphrasing with a Pivot-Language

Another relevant line of research is on using multilingual parallel corpora to improve SMT using additional languages as pivots.

Callison-Burch et al. (2006) improved English→Spanish and English→French SMT using source-language paraphrases extracted with the pivoting technique of Bannard and Callison-Burch (2005) and eight additional languages from the *Europarl corpus* (Koehn, 2005). For example, using German as a pivot, they extracted English paraphrases from a parallel English-German corpus by looking for English phrases that were aligned to the same German phrase: *e.g.*, if *under control* and *in check* were aligned to *unter controlle*, they were hypothesized to be paraphrases with some probability. Such (English) paraphrases were added as additional entries in the phrase table of an English→Spanish/English→French phrase-based SMT system and paired with the foreign (Spanish/French) translation of the original (English) phrase. The system was then tuned with MERT using an extra feature penalizing low-probability paraphrases; this yielded up to 1.8% absolute improvement in Bleu.

Other important publications about pivoting approaches for machine translation include (Wu and Wang, 2007), (Utiyama and Isahara, 2007), (Hajič et al., 2000) and (Habash and Hu, 2009).

Unlike pivoting, which can only improve source-language lexical coverage, we augment both the source- and the target-language sides. Second, while pivoting ignores context when extracting paraphrases, we take it into account. Third, by using as an additional language one that is related to the source, we are able to get increase in Bleu that is comparable and even better than what pivoting achieves with eight pivot languages. On the negative side, our approach is limited in that it requires that X_2 be related to X_1, while the pivoting language Z does not need to be related to X_1 nor to Y. However, we only need one additional parallel corpus (for X_2-Y), while pivoting needs two: one for X_1-Z and one for Z-Y. Finally, note that our approach is orthogonal to pivoting, and thus the two can be combined.

6 Conclusion and Future Work

We have proposed a novel method for improving SMT for resource-poor languages by exploiting their similarity to resource-rich ones.

In future work, we would like to extend that approach in several interesting directions. First, we want to make better use of multi-lingual parallel corpora, *e.g.*, while we had access to a Spanish-Portuguese-English corpus, we used it as two separate bi-texts Spanish-English and Portuguese-English. Second, we would like to exploit multiple auxiliary resource-rich languages the resource-poor source language is related to. Third, we could also experiment with using auxiliary languages that are related to the *target* language.

Acknowledgments

This research was supported by research grant POD0713875.

References

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz Joseph Och, David Purdy, Noah Smith, and David Yarowsky. 1999. Statistical machine translation. Technical report, CLSP, Johns Hopkins University, Baltimore, MD.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of ACL'05*, pages 597–604.

Shane Bergsma and Grzegorz Kondrak. 2007. Alignment-based discriminative string similarity. In *Proceedings of ACL'07*, pages 656–663.

Albert Bickford and David Tuggy. 2002. Electronic glossary of linguistic terms. http://www.sil.org/mexico/ling/glosario/E005ai-Glossary.htm.

Alexandra Birch, Miles Osborne, and Philipp Koehn. 2007. CCG supertags in factored statistical machine translation. In *Proceedings of WMT'2007*, pages 9–16.

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

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of HLT-NAACL'06*, pages 17–24.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL'05*, pages 531–540.

Andrew Finch and Eiichiro Sumita. 2008. Phrase-based machine transliteration. In *Proceedings of WTCAST'08*, pages 13–18.

Nizar Habash and Jun Hu. 2009. Improving Arabic-Chinese statistical machine translation using English as pivot language. In *Proceedings of the WMT'09*, pages 173–181.

Jan Hajič, Jan Hric, and Vladislav Kuboň. 2000. Machine translation of very close languages. In *Proceedings of ANLP'00*, pages 7–12.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of IWSLT'05*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL'07. Demonstration session*, pages 177–180.

Philipp Koehn. 2005. Europarl: A parallel corpus for evaluation of machine translation. In *Proceedings of MT Summit*, pages 79–86.

Grzegorz Kondrak, Daniel Marcu, and Kevin Knight. 2003. Cognates can improve statistical translation models. In *Proceedings of NAACL'03*, pages 46–48.

Gideon Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *Proceedings of NAACL'01*, pages 1–8.

Dan Melamed. 1995. Automatic evaluation and uniform filter cascades for inducing N-best translation lexicons. In *Proceedings of WVLC'95*, pages 184–198.

Dan Melamed. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25(1):107–130.

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

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

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL'03*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL'02*, pages 311–318.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Daniel Varga. 2006. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of LREC'2006*, pages 2142–2147.

Jorg Tiedemann. 1999. Automatic construction of weighted string similarity measures. In *Proceedings of EMNLP-VLC'99*, pages 213–219.

Masao Utiyama and Hitoshi Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *Proceedings of NAACL-HLT'07*, pages 484–491.

Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. *Machine Translation*, 21(3):165–181.

What's in a name? In some languages, grammatical gender

Vivi Nastase
EML Research gGmbH
Heidelberg, Germany
nastase@eml-research.de

Marius Popescu
Department of Mathematics and Computer Science
University of Bucharest, Bucharest, Romania
mpopescu@phobos.cs.unibuc.ro

Abstract

This paper presents an investigation of the relation between words and their gender in two gendered languages: German and Romanian. Gender is an issue that has long preoccupied linguists and baffled language learners. We verify the hypothesis that gender is dictated by the general sound patterns of a language, and that it goes beyond suffixes or word endings. Experimental results on German and Romanian nouns show strong support for this hypothesis, as gender prediction can be done with high accuracy based on the form of the words.

1 Introduction

For speakers of a language whose nouns have no gender (such as modern English), making the leap to a language that does (such as German), does not come easy. With no or few rules or heuristics to guide him, the language learner will try to draw on the "obvious" parallel between grammatical and natural gender, and will be immediately baffled to learn that *girl – Mädchen* – is neuter in German. Furthermore, one may refer to the same object using words with different gender: *car* can be called *(das) Auto* (neuter) or *(der) Wagen* (masculine). Imagine that after hard work, the speaker has mastered gender in German, and now wishes to proceed with a Romance language, for example Italian or Spanish. He is now confronted with the task of relearning to assign gender in these new languages, made more complex by the fact that gender does not match across languages: e.g. *sun* – feminine in German (*die Sonne*), but masculine in Spanish (*el sol*), Italian (*il sole*) and French (*le soleil*); *moon* – masculine in German (*der Mond*), but feminine in Spanish (*la luna*), Italian (*la luna*) and French (*la lune*). Gender doesn't even match within a single language family: *travel* – masculine in Spanish (*el viage*) and Italian (*il viaggio*), but feminine in Portuguese (*a viagem*).

Grammatical gender groups nouns in a language into distinct classes. There are languages whose nouns are grouped into more or less than three classes. English for example has none, and makes no distinction based on gender, although Old English did have three genders and some traces remain (e.g. *blonde, blond*).

Linguists assume several sources for gender: (i) a first set of nouns which have natural gender and which have associated matching grammatical gender; (ii) nouns that resemble (somehow) the nouns in the first set, and acquire their grammatical gender through this resemblance. Italian and Romanian, for example, have strong and reliable phonological correlates (Vigliocco et al., 2004b, for Italian). (Doca, 2000, for Romanian). In Romanian the majority of feminine nouns end in *ă* or *e*. Some rules exists for German as well (Schumann, 2006), for example nouns ending in *-tät, -ung, -e, -enz, -ur, -keit, -in* tend to be feminine. Also, when specific morphological processes apply, there are rules that dictate the gender of the newly formed word. This process explains why *Frau* (woman) is feminine in German, while *Fräulein* (little woman, miss) is neuter – *Fräulein = Frau + lein*. The existing rules have exceptions, and there are numerous nouns in the language which are not derived, and such suffixes do not apply.

Words are names used to refer to concepts. The fact that the same concept can be referred to using names that have different gender – as is the case for *car* in German – indicates that at least in some cases, grammatical gender is in the name and not the concept. We test this hypothesis – that the gender of a noun is in its word form, and that this goes beyond word endings – using noun gender data for German and Romanian. Both Romanian and German have 3 genders: masculine, feminine and

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1368–1377,
Singapore, 6-7 August 2009.

neuter. The models built using machine learning algorithms classify test nouns into gender classes based on their form with high accuracy. These results support the hypothesis that in gendered languages, the word form is a strong clue for gender. This supplements the situation when some concepts have natural gender that matches their grammatical gender: it allows for an explanation where there is no such match, either directly perceived, or induced through literary devices.

The present research has both theoretical and practical benefits. From a theoretical point of view, it contributes to research on phonology and gender, in particular in going a step further in understating the link between the two. From a practical perspective, such a connection between gender and sounds could be exploited in advertising, in particular in product naming, to build names that fit a product, and which are appealing to the desired customers. Studies have shown that especially in the absence of meaning, the form of a word can be used to generate specific associations and stimulate the imagination of prospective customers (Sells and Gonzales, 2003; Bedgley, 2002; Botton et al., 2002).

2 Gender

What is the origin of grammatical gender and how does it relate to natural gender? Opinions are split. Historically, there were two main, opposite, views: (i) there is a semantic explanation, and natural gender motivated the category (ii) the relationship between natural and grammatical gender is arbitrary.

Grimm (1890) considered that grammatical gender is an extension of natural gender brought on by imagination. Each gender is associated with particular adjectives or other attributes, and in some cases (such as for *sun* and *moon*) the assignment of gender is based on personification. Brugmann (1889) and Bloomfield (1933) took the position that the mapping of nouns into genders is arbitrary, and other phenomena – such as derivations, personification – are secondary to the established agreement. Support for this second view comes also from language acquisition: children who learn a gendered language do not have a natural gender attribute that they try to match onto the newly acquired words, but learn these in a separate process. Any match or mapping between natural and grammatical gender is done after the natural gender "feature" is acquired itself. Kilarski (2007) presents a more detailed overview of currents and ideas about the origin of gender. Unterbeck (1999) contains a collection of papers that investigate grammatical gender in several languages, aspects of gender acquisition and its relation with grammatical number and agreement.

There may be several reasons for the polemic between these two sides. One may come from the categorization process, the other from the relation between word form and its meaning. Let us take them each in turn, and see how they influenced gender.

Grammatical gender separates the nouns in a language into disjoint classes. As such, it is a categorization process. The traditional – classical – theory of categorization and concepts viewed categories and concepts as defined in terms of a set of common properties that all its members should share. Recent theories of concepts have changed, and view concepts (and categories) not necessarily as "monolithic" and defined through rules, but rather as clusters of members that may resemble each other along different dimensions (Margolis and Laurence, 1999).

In most linguistic circles, the principle of arbitrariness of the association between form and meaning, formalized by de Saussure (1916) has been largely taken for granted. It seems however, that it is hard to accept such an arbitrary relation, as there have always been contestants of this principle, some more categorical than others (Jakobson, 1937; Jespersen, 1922; Firth, 1951). It is possible that the correlation we perceive between the word form and the meaning is something that has arisen after the word was coined in a language, being the result of what Firth called "phonetic habit" through "an attunement of the nervous system", and that we have come to prefer, or select, certain word forms as more appropriate to the concept they name – "There is no denying that there are words which we feel instinctively to be adequate to express the ideas they stand for. ... Sound symbolism, we may say, makes some words more fit to survive" (Jespersen, 1922).

These two principles relate to the discussion on gender in the following manner: First of all, the categories determined by grammatical gender need not be homogeneous, and their members need not all respect the same membership criterion. This frees us from imposing a matching between natural and grammatical gender where no such relation is obvious or pos-

sible through literary devices (personification, metaphor, metonymy). Nouns belonging to the same gender category may resemble each other because of semantic considerations, lexical derivations, internal structure, perceived associations and so on. Second, the fact that we allow for the possibility that the surface form of a word may encode certain word characteristics or attributes, allows us to hypothesize that there is a surface, phonological, similarity between words grouped within the same gender category, that can supplement other resemblance criteria in the gender category (Zubin and Köpcke, 1986).

Zubin and Köpcke (1981), Zubin and Köpcke (1986) have studied the relation between semantic characteristics and word form with respect to gender for German nouns. Their study was motivated by two observations: Zipf (1935) showed that word length is inversely correlated with frequency of usage, and Brown (1958) proposed that in choosing a name for a given object we are more likely to use a term corresponding to a "basic" level concept. For example, *chair, dog, apple* would correspond to the basic level, while *furniture, animal, fruit* and *recliner, collie, braeburn apple* correspond to a more general or a more specific level, respectively. Their study of gender relative to these levels have shown that basic level terms have masculine, feminine, and rarely neuter genders, while the more undifferentiated categories at the superordinate level are almost exclusively neuter.

In psycholinguistic research, Friederici and Jacobsen (1999) adopt the position that a lexical entry consists of two levels: form and semantic and grammatical properties to study the influence of gender priming – both from a form and semantic perspective – on language comprehension. Vigliocco et al. (2004a) study gender priming for German word production. While this research studies the influence of the word form on the production of nouns with the same or different grammatical gender, there is no study of the relation between word forms and their corresponding gender.

In recent studies we have found on the relation between word form and its associated gender, the only phonological component of a word that is considered indicative is the ending. Spalek et al. (2008) experiment on French nouns, and test whether a noun's ending is a strong clue for gender for native speakers of French. Vigliocco et al. (2004b) test cognitive aspects of grammatical gender of Italian nouns referring to animals.

Cucerzan and Yarowsky (2003) present a bootstrapping process to predict gender for nouns in context. They show that context gives accurate clues to gender (in particular through determiners, quantifiers, adjectives), but when the context is not useful, the algorithm can fall back successfully on the word form. Cucerzan and Yarowsky model the word form for predicting gender using suffix trie models. When a new word is encountered, the word is mapped onto the trie starting from the last letter, and it is assigned the gender that has the highest probability based on the path it matches in the trie. In context nouns appear with various inflections – for number and case in particular. Such morphological derivations are gender specific, and as such are strong indicators for gender.

The hypothesis tested here is that gender comes from the general sound of the language, and is distributed throughout the word. For this, the data used should not contain nouns with "tell tale" inflections. The data will therefore consist of nouns in the singular form, nominative case. Some nouns are derived from verbs, adverbs or adjectives, or other nouns through morphological derivations. These derivations are regular and are identifiable through a rather small number of regular suffixes. These suffixes (when they are indicative of gender) and word endings will be used as baselines to compare the accuracy of prediction on the full word with the ending fragment.

3 Data

We test our gender-language sounds connection through two languages from different language families. German will be the representative of the Germanic languages, and Romanian for the Romance ones. We first collect data in the two languages, and then represent them through various features – letters, pronunciation, phonetic features.

3.1 Noun collections

German data For German we collect nouns and their grammatical gender from a German-English dictionary, part of the BEOLINGUS multi-lingual dictionary[1]. In the first step we collected the German nouns and their gender from this dictionary. In step 2, we filter out compounds. The reason for this step is that a German noun compound will

[1] `http://dict.tu-chemnitz.de/`

have the gender of its head, regardless of its nominal modifiers. For the lack of a freely available tool to detect and split noun compounds, we resort to the following algorithm:

1. initialize the list of nouns L_N to the empty list;
2. take each noun n in the dictionary D, and
   (a) if $\exists n_i \in L_N$ such that n is an end substring of n_i, then add n to L_N and remove n_i from L_N;
   (b) if $\exists n_i \in L_N$ such that n_i is a end substring of n, skip n;

Essentially, we remove from the data all nouns that include another noun as the end part (which is the head position in German noun compounds). This does not filter examples that have suffixes added to form the feminine version of a masculine noun, for example: *(der) Lehrer – (die) Lehrerin* (teacher). The suffixes are used in one of the baselines for comparison with our learning method.

We obtain noun pronunciation information from the Bavarian Archive for Speech Signals[2]. We filter again our list L_N to keep nouns for which we have pronunciation information. This allows us to compare the learning results when letter or pronunciation information is used.

After collecting the nouns and their pronunciation, we map the pronunciation onto lower level phonetic features, following the IPA encoding of sounds for the German language. The mapping between sounds and IPA features was manually encoded following IPA tables.

Romanian data We extract singular nominative forms of nouns from the Romanian lexical database (Barbu, 2008). The resource contains the proper word spelling, including diacritics and special characters. Because of this and the fact that there is a straightforward mapping between spelling and pronunciation in Romanian, we can use the entire data extracted from the dictionary in our experiments, without special pronunciation dictionaries. Following the example for the German language, we encode each sound through lower level phonological features using IPA guidelines.

As in Italian, in Romanian there are strong phonological cues for nouns, especially those having the feminine gender: they end in *ă* and *e*. To determine whether the connection between a word form and gender goes beyond this superficial rule, we generate a dataset in which the nouns are stripped of their final letter, and their representation is built based on this reduced form.

Table 1 shows the data collected and the distribution in the three classes.

	German		Romanian	
masc.	565	32.64%	7338	15.14%
fem.	665	38.42%	27187	56.08%
neut.	501	28.94%	13952	28.78%
total	1731		48477	

Table 1: Data statistics

Because for Romanian the dataset is rather large, we can afford to perform undersampling to balance our classes, and have a more straightforward evaluation. We generate a perfectly balanced dataset by undersampling the feminine and the neuter classes down to the level of the masculine class. We work then with a dataset of 22014 instances, equally distributed among the three genders.

3.2 Data representation

For each word in our collections we produce three types of representation: letters, phonemes and phonological features. Table 2 shows examples for each of these representations. The letter and phoneme representations are self-explanatory. We obtain the pronunciation corresponding to each word from a pronunciation dictionary, as mentioned in Section 3.1, which maps a word onto a sequence of phonemes (phones). For Romanian we have no such resource, but me make without since in most part the pronunciation matches the letter representation[3].

German		
letter	abend (m)	a b e n d
phoneme		a: b @ n d
Romanian		
letter	seară (f)	s e a r ă

Table 2: Data representation in terms of letters and phonemes for the German and Romanian forms of the word *evening*. For Romanian, the letter and phoneme representation is the same.

[2] http://www.phonetik.uni-muenchen.de/Bas/

[3] The exceptions are the diphthongs and a few groups of letters: ce, ci, che, chi, oa, and the letter x.

Phonemes, the building blocks of the phonetic representation, can be further described in terms of phonological features – "configurations" of the vocal tract (e.g tongue and lips position), and acoustic characteristics (e.g. manner of air flow). We use IPA standards for mapping phones in German and Romanian onto these phonological features. We manually construct a map between phones and features, and then automatically binarize this representation and use it to generate a representation for each phone in each word in the data. For the word *abend (de) / seara (ro) (evening)* in Figure 2, the phonological feature representation for German is:

0000100000001000010000000001
0001000100000000000010000000
0000100000000100000000010001
1000000100000010000000000000
1000000100000000000010000000,

with the feature base:

< alveolar, approximant, back, bilabial, central, close, closemid, consonant, fricative, front, glottal, labiodental, long, mid, nasal, nearclose, nearopen, open, openmid, palatal, plosive, postalveolar, rounded, short, unrounded, uvular, velar, vowel >.

For Romanian, the phonological feature base is:

< accented, affricate, approximant, back, bilabial, central, close, consonant, dental, fricative, front, glottal, labiodental, mid, nasal, open, plosive, postalveolar, rounded, trill, unrounded, velar, voiced, voiceless, vowel >,

and the phonological feature representation of the word changes accordingly.

4 Kernel Methods and String Kernels

Our hypothesis that the gender is in the name is equivalent to proposing that there are sequences of letters/sounds/phonological features that are more common among nouns that share the same gender or that can distinguish between nouns under different genders. To determine whether that is the case, we use a string kernel, which for a given string (sequence) generates a representation that consists of all its substrings of length less than a parameter l. The words are represented as strings with boundaries marked with a special character ('#'). The high dimensional representation generated by the string kernel is used to find a hyperplane that separates instances of different classes. In this section we present in detail the kernel we use.

Kernel-based learning algorithms work by embedding the data into a feature space (a Hilbert space), and searching for linear relations in that space. The embedding is performed implicitly, that is by specifying the inner product between each pair of points rather than by giving their coordinates explicitly.

Given an input set $\mathcal{X}$ (the space of examples), and an embedding vector space $\mathcal{F}$ (feature space), let $\phi : \mathcal{X} \rightarrow \mathcal{F}$ be an embedding map called feature map.

A *kernel* is a function k, such that for all $x, z \in \mathcal{X}$, $k(x, z) =< \phi(x), \phi(z) >$, where $< ., . >$ denotes the inner product in $\mathcal{F}$.

In the case of binary classification problems, kernel-based learning algorithms look for a discriminant function, a function that assigns $+1$ to examples belonging to one class and -1 to examples belonging to the other class. This function will be a linear function in the space $\mathcal{F}$, that means it will have the form:

$$f(x) = \text{sign}(< w, \phi(x) > +b),$$

for some weight vector w. The kernel can be exploited whenever the weight vector can be expressed as a linear combination of the training points, $\sum_{i=1}^{n} \alpha_i \phi(x_i)$, implying that f can be expressed as follows:

$$f(x) = \text{sign}(\sum_{i=1}^{n} \alpha_i k(x_i, x) + b)$$

.

Various kernel methods differ in the way in which they find the vector w (or equivalently the vector α). Support Vector Machines (SVM) try to find the vector w that define the hyperplane that maximum separate the images in $\mathcal{F}$ of the training examples belonging to the two classes. Mathematically SVMs choose the w and b that satisfy the following optimization criterion:

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i(< w, \phi(x_i) > +b)]_+ + \nu ||w||^2$$

where y_i is the label ($+1/-1$) of the training example x_i, ν a regularization parameter and $[x]_+ = \max(x, 0)$.

Kernel Ridge Regression (KRR) selects the vector w that simultaneously has small empirical error and small norm in Reproducing Kernel Hilbert Space generated by kernel k. The resulting minimization problem is:

$$\min_w \frac{1}{n}\sum_{i=1}^{n}(y_i - < w, \phi(x_i) >)^2 + \lambda||w||^2$$

where again y_i is the label ($+1/-1$) of the training example x_i, and λ a regularization parameter. Details about SVM and KRR can be found in (Taylor and Cristianini, 2004). What is important is that above optimization problems are solved in such a way that the coordinates of the embedded points are not needed, only their pairwise inner products which in turn are given by the kernel function k.

SVM and KRR produce binary classifiers and gender classification is a multi-class classification problem. There are a lot of approaches for combining binary classifiers to solve multi-class problems. We used *one-vs-all* scheme. For arguments in favor of one-vs-all see (Rifkin and Klautau, 2004).

The kernel function offers to the kernel methods the power to naturally handle input data that are not in the form of numerical vectors, for example strings. The kernel function captures the intuitive notion of similarity between objects in a specific domain and can be any function defined on the respective domain that is symmetric and positive definite. For strings, a lot of such kernel functions exist with many applications in computational biology and computational linguistics (Taylor and Cristianini, 2004).

Perhaps one of the most natural ways to measure the similarity of two strings is to count how many substrings of length p the two strings have in common. This give rise to the p-spectrum kernel. Formally, for two strings over an alphabet Σ, $s, t \in \Sigma^*$, the p-spectrum kernel is defined as:

$$k_p(s,t) = \sum_{v \in \Sigma^p} \text{num}_v(s)\text{num}_v(t)$$

where $\text{num}_v(s)$ is the number of occurrences of string v as a substring in s [4] The feature map defined by this kernel associate to each string a vector of dimension $|\Sigma|^p$ containing the histogram of frequencies of all its substrings of length p. Taking into account all substrings of length less than p it will be obtained a kernel that is called the *blended spectrum kernel*:

$$k_1^p(s,t) = \sum_{q=1}^{p} k_q(s,t)$$

The blended spectrum kernel will be the kernel that we will use in conjunction with SVM and KRR. More precisely we will use a normalized version of the kernel to allow a fair comparison of strings of different length:

$$\hat{k}_1^p(s,t) = \frac{k_1^p(s,t)}{\sqrt{k_1^p(s,s)k_1^p(t,t)}}$$

5 Experiments and Results

We performed 10-fold cross-validation learning experiments with kernel ridge regression and the string kernel (KRR-SK) presented in Section 4. We used several baselines to compare the results of the experiments against:

BL-R Gender is assigned following the distribution of genders in the data.

BL-M Gender is assigned following the majority class (only for German, for Romanian we use balanced data).

BL-S Gender is assigned based on suffix-gender relation found in the literature. We use the following mappings:

- German (Schumann, 2006):
  feminine *-ade, -age, -anz, -e, -ei, -enz, -ette, -heit, -keit, -ik, -in, -ine, -ion, -itis, -ive, -schaft, -tät, -tur, -ung, -ur*;
  masculine *-ant, -er, -ich, -ismus, -ling*;
  neuter *-chen, -ist, -lein, -ment, -nis, -o, -tel, -um*.
  In our data set the most dominant gender is feminine, therefore we assign this gender to all nouns that do not match any of the previous suffixes. Table 4 shows a few suffixes for each gender, and an example noun.
- Romanian: in Romanian the word ending is a strong clue for gender, especially for feminine nouns: the vast majority end in either *-e* or *-ă* (Doca, 2000). We design a heuristic that assigns the gender "preferred" by the last letter – the

[4] Note that the notion of substring requires contiguity. See (Taylor and Cristianini, 2004) for discussion about the ambiguity between the terms "substring" and "subsequence" across different traditions: biology, computer science.

Method	Accuracy	masc. F-score	fem. F-score	neut. F-score
German				
BL-R	33.79			
BL-M	38.42			
BL-S	51.35	40.83	62.42	26.69
KRR-SK	72.36 ± 3	64.88 ± 5	84.34 ± 4	64.44 ± 7
KRR-SK$_{noWB}$	66.91	58.77	79.19	58.26
Romanian				
BL-R	33.3			
BL-S	74.38	60.65	97.96	63.93
KRR-SK	78.83 ± 0.8	68.74 ± 0.9	98.05 ± 0.2	69.38 ± 2
KRR-SK no last letter	65.73 ± 0.6	56.11 ± 1	85.00 ± 0.5	55.05 ± 1
KRR-SK$_{noWB}$	77.36	67.54	96.75	67.39

Table 3: 10-fold cross-validation results – accuracy and f-scores percentages ($\pm$ variation over the 10 runs) – for gender learning using string kernels

German

gender	suffix	example	
fem.	-e	Ecke	(corner)
	-heit	Freiheit	(freedom)
	-ie	Komödie	(comedy)
masc.	-er	Fahrer	(driver)
	-ich	Rettich	(radish)
	-ling	Frühling	(spring - season)
neut.	-chen	Mädchen	(girl)
	-nis	Verständnis	(understanding)
	-o	Auto (car)	

Table 4: Gender assigning rules and examples for German

Romanian

gender	ending	example		Prec.
fem.	-ă	masă	(table)	98.04
	-e	pâine	(bread)	97.89
masc.	-g	sociolog	(sociologist)	72.77
	-r	nor	(cloud)	66.89
	-n	domn	(gentleman)	58.45
neut.	-m	algoritm	(algorithm)	90.95
	-s	vers	(verse)	66.97
	-t	eveniment	(event)	51.02

Table 5: Word-ending precision on classifying gender and examples for Romanian

majority gender of all nouns ending in the respective letter – based on analysis of our data. In Table 5 we include some of the letter endings with an example noun, and a percentage that shows the precision of the ending in classifying the noun in the gender indicated in the table.

The results of our experiments are presented in Table 3, in terms of overall accuracy, and f-score for each gender. The performance presented corresponds to the letter-based representation of words. It is interesting to note that this representation performed overall better than the phoneme or phonological feature-based ones. An explanation may be that in both the languages we considered, there is an (almost) one-to-one mapping between letters and their pronunciation, making thus the pronunciation-based representation unnecessary. As such, the letter level captures the interesting commonalities, without the need to go down to the phoneme-level.

We performed experiments for Romanian when the last letter of the word is removed. The reason for this batch of experiments is to further test the hypothesis that gender is more deeply encoded in a word form than just the word ending. For both languages we observe statistically significant higher performance than all baselines. For Romanian, the last letter heuristic gives a very high baseline, confirming that Romanian has strong phonological cues for gender in the ending. Had the word ending been the only clue to the word's gender,

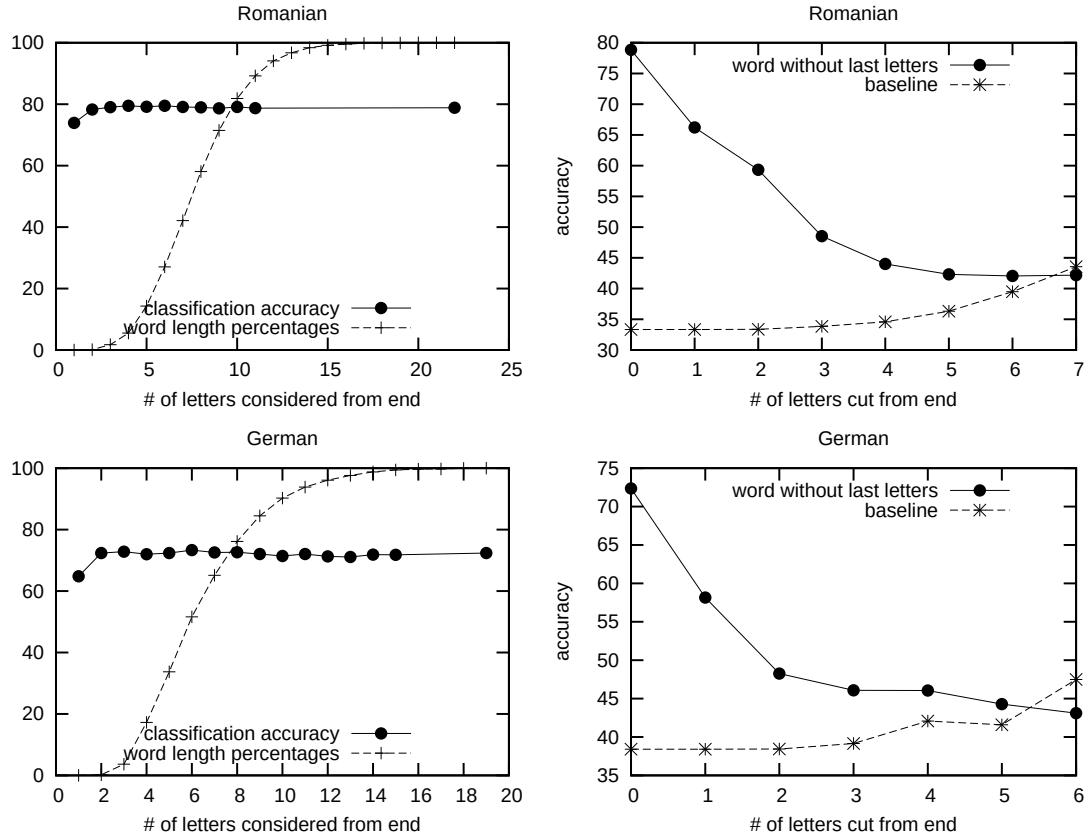


Figure 1: Gender prediction based on the last N letters, and based on the word minus the last N letters

once it is removed the performance on recognizing gender should be close to the random assignment. This is not the case, and the improvement over the random baseline is 32% points. It is interesting to notice that when cutting off the last letter the class for which the gender assignment heuristic was clearest – the feminine class with -ă and -e endings – the performance remains very high – 85% F-score.

To further test where the gender indicators are located, we performed two more sets of experiments: (i) classify words in their corresponding gender class using the word minus the last N letters; (ii) classify words based on the last N letters. The results of these experiments in terms of accuracy are presented in Figure 1. When considering only the last N letters the performance is high for both German and Romanian, as expected if the gender indicators are concentrated at the end of the word. It is interesting though to notice the results of classification based on the word without the last N letters. The prediction accuracy monotonically decreases, but remains above the baseline until more than 6 letters are cut. Because as letters are cut some words completely disappear, the baseline changes accordingly. 94.07% of the words have a length of at most 12 letters in the Romanian dataset, and 96.07% in the German one. Because gender prediction can be done with accuracy higher than the random baseline even after 6 letters are cut from the ending of the word indicate that for more than 94% of the words considered, gender clues are spread over more than the second half of the word. Again, we remind the reader that the word forms are in nominative case, with no case or number inflections (which are strong indicators of gender in both Romanian and German).

Except for lines $KRR - SK_{noWB}$, the results in Table 3 are obtained through experiments conducted on words containing word boundary markers, as indicated in Section 4. Because of these markers, word starting or word ending substrings are distinct from all the others, and information about their position in the original word is thus preserved. To further explore the idea that gender indicators are not located only in word endings, we ran classification experiments for German and Romanian when the word representation does not contain word boundary markers. This means that the substrings generated by the string kernel have

no position information. The results of these experiments are presented in rows $KRR-SK_{noWB}$ in Table 3. The accuracy is slightly lower than the best results obtained when word boundaries are marked and the entire word form is used. However, they are well above all the baselines considered, without no information about word endings.

For both German and Romanian, the gender that was learned best was feminine. For German part of this effect is due to the fact that the feminine class is more numerous in the data. For Romanian the data was perfectly balanced, so there is no such bias. Neuter and masculine nouns have lower learning performance. For Romanian, a contribution to this effect is the fact that neuter nouns behave as masculine nouns in their singular form (take the same articles, inflections, derivations), but as feminine in the plural, and our data consists of nouns in singular form. It would seem that from an orthographic point of view, neuter and masculine nouns are closer to each other than to feminine nouns.

From the reviewed related work, the one that uses the word form to determine gender is Cucerzan and Yarowsky (2003) for Romanian. There are two important differences with respect to the approach presented here. First, they consider words in context, which are inflected for number and case. Number and case inflections are reflected in suffixes that are gender specific. The words considered here are in singular form, nominative case – as such, with no inflections. Second, Cucerzan and Yarowsky consider two classes: feminine vs. masculine and neuter. Masculine and neuter nouns are harder to distinguish, as in singular form neuter nouns behave like masculine nouns in Romanian. While the datasets and word forms used by Cucerzan and Yarowsky are different than the one used here, the reader may be curious how well the word form distinguishes between feminine and the other two classes in the experimental set-up used here. On the full[5] Romanian dataset described in Section 3, a two class classification gives 99.17% accuracy. When predicting gender for all words in their dataset, Cucerzan and Yarowsky obtain 98.25% accuracy.

6 Conclusion

When a speaker of a genderless language tries to learn a language with grammatical gender, it is very tempting to try to assign grammatical gender based on perceived or guessed natural gender types. This does not work out well, and it only serves to confuse the learner even more, when he finds out that nouns expressing concepts with clear feminine or masculine natural gender will have the opposite or a neutral grammatical gender, or that one concept can be referred to through names that have different grammatical genders. Going with the flow of the language seems to be a better idea, and allow the sound of a word to dictate the gender.

In this paper we have investigated the hypothesis that gender is encoded in the word form, and this encoding is more than just the word endings as it is commonly believed. The results obtained show that gender assignment based on word form analysis can be done with high accuracy – 72.36% for German, and 78.83% for Romanian. Existing gender assignment rules based on word endings have lower accuracy. We have further strengthened the point by conducting experiments on Romanian nouns without tell-tale word endings. The accuracy remains high, with remarkably high performance in terms of F-score for the feminine class (85%). This leads us to believe that gender information is somehow redundantly coded in a word. We plan to look closer at cases where we obtain different predictions based on the word ending and the full form of the word, and use boosting to learn weights for classifiers based on different parts of the word to see whether we can further improve the results.

As we have underlined before, word form similarity between words under the same gender is one criterion for gender assignment. It would be interesting to verify whether gender recognition can be boosted by using lexical resources that capture the semantics of the words, such as WordNets or knowledge extracted from Wikipedia, and verify whether similarities from a semantic point of view are also responsible for gender assignments in various languages.

[5] By "full" we mean the dataset before balancing the classes 48,477 instances (see Table 1).

References

Ana-Maria Barbu. 2008. Romanian lexical data bases: Inflected and syllabic forms dictionaries. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*. http://www.lrec-conf.org/proceedings/lrec2008/.

Sharon Bedgley. 2002. Strawberry is no blackberry: Building brands us-

ing sound. http://online.wsj.com/article/0,,SB10303107301794746 75.djm,00.html.

Leonard Bloomfield. 1933. *Language*. Holt, Reinhart & Winston, New York.

Marcel Botton, Jean-Jack Cegarra, and Beatrice Ferrari. 2002. *Il nome della marca: creazione e strategia di naming, 3rd edition*. Guerini e Associati.

Roger Brown. 1958. *Words and Things*. The Free Press, New York.

Karl Brugmann. 1889. Das Nominalgeschlecht in den indogermanischen Sprachen. In *Internationale Zeitschrift für allgemenine Sprachwissenschaft*, pages 100–109.

S. Cucerzan and D. Yarowsky. 2003. Minimally supervised induction of grammatical gender. In *Proceedings of HLT-NAACL 2003*, pages 40–47.

Ferdinand de Saussure. 1916. *Cours de linguistique générale*. Harrassowitz, Wiesbaden.

Gheorghe Doca Doca. 2000. *Romanian language. Vol. II: Morpho-Syntactic and Lexical Structures*. Ars Docendi, Bucharest, Romania.

John Rupert Firth. 1951. Modes and meaning. In *Papers in linguistics 1934-1951*. Oxford University Press, London.

Angela Friederici and Thomas Jacobsen. 1999. Processing grammatical gender during language comprehension. *Journal of Psychological Research*, 28(5):467–484.

Jacob Grimm. 1890. *Deutsche Grammatik*.

Roman Jakobson. 1937. *Lectures on Sound and Meaning*. MIT Press, Cambridge, MA.

Otto Jespersen. 1922. *Language - its Nature, Development and Origin*. George Allen & Unwim Ltd., London.

Marcin Kilarski. 2007. On grammatical gender as an arbitrary and redundant category. In Douglas Kilbee, editor, *History of Linguistics 2005: Selected papers from the 10th International Conference on the History of Language Sciences (ICHOLS X)*, pages 24–36. John Benjamins, Amsterdam.

Eric Margolis and Stephen Laurence, editors. 1999. *Concepts: Core Readings*. MIT Press.

Ryan Rifkin and Aldebaro Klautau. 2004. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5(January):101–141.

Johannes Schumann. 2006. *Mittelstufe Deutsch*. Max Hueber Verlag.

Peter Sells and Sierra Gonzales. 2003. The language of advertising. http://www.stanford.edu/class/linguist34/; in particular unit 8: ˜/Unit_08/blackberry.htm.

Katharina Spalek, Julie Franck, Herbert Schriefers, and Ulrich Frauenfelder. 2008. Phonological regularities and grammatical gender retrieval in spoken word recognition and word production. *Journal of Psycholinguistic Research*, 37(6):419–442.

John S. Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.

Barbara Unterbeck, editor. 1999. *Gender in Grammar and Cognition. Approaches to Gender*. Trends in Linguistics. Studies and Monographs. 124. Mouton de Gruyter.

Gabriela Vigliocco, David Vinson, Peter Indefrey, Willem Levelt, and Frauke Hellwig. 2004a. Role of grammatical gender and semantics in german word production. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 30(2):483–497.

Gabriela Vigliocco, David Vinson, and Federica Paganelli. 2004b. Grammatical gender and meaning. In *Proc. of the 26th Meeting of the Cognitive Science Society*.

George Zipf. 1935. *The Psychobiology of Language*. Addison-Wesley.

David Zubin and Klaus-Michael Köpcke. 1981. Gender: A less than arbitrary grammatical category. In R. Hendrick, C. Masek, and M. F. Miller, editors, *Papers from the seventh regional meeting*, pages 439–449. Chicago Linguistic Society, Chicago.

David Zubin and Klaus-Michael Köpcke. 1986. Gender and folk taxonomy: The indexical relation between grammatical and lexical categorization. In C. Craig, editor, *Noun classes and categorization*, pages 139–180. Benjamins, Philadelphia.

Convolution Kernels on Constituent, Dependency and Sequential Structures for Relation Extraction

Truc-Vien T. Nguyen and **Alessandro Moschitti** and **Giuseppe Riccardi**
nguyenthi,moschitti,riccardi@disi.unitn.it
Department of Information Engineering and Computer Science
University of Trento
38050 Povo (TN), Italy

Abstract

This paper explores the use of innovative kernels based on syntactic and semantic structures for a target relation extraction task. Syntax is derived from constituent and dependency parse trees whereas semantics concerns to entity types and lexical sequences. We investigate the effectiveness of such representations in the automated relation extraction from texts. We process the above data by means of Support Vector Machines along with the syntactic tree, the partial tree and the word sequence kernels. Our study on the ACE 2004 corpus illustrates that the combination of the above kernels achieves high effectiveness and significantly improves the current state-of-the-art.

1 Introduction

Relation Extraction (RE) is defined in ACE as the task of finding relevant semantic relations between pairs of entities in texts. Figure 1 shows part of a document from ACE 2004 corpus, a collection of news articles. In the text, the relation between *president* and *NBC's entertainment division* describes the relationship between the first entity (person) and the second (organization) where the person holds a managerial position.

Several approaches have been proposed for automatically learning semantic relations from texts. Among others, there has been increased interest in the application of kernel methods (Zelenko et al., 2002; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005a; Bunescu and Mooney, 2005b; Zhang et al., 2005; Wang, 2008). Their main property is the ability of exploiting a huge amount of features without an explicit feature representation. This can be done by computing a kernel function between a pair of linguistic objects, where such function is a kind of similarity measure satisfying certain properties. An example is the sequence kernel (Lodhi et al., 2002), where the objects are strings of characters and the kernel function computes the number of common subsequences of characters in the two strings. Such substrings are then weighted according to a decaying factor penalizing longer ones. In the same line, Tree Kernels count the number of *subtree* shared by two input trees. An example is that of syntactic (or subset) tree kernel (SST) (Collins and Duffy, 2001), where trees encode grammatical derivations.

> Jeff Zucker, the longtime executive producer of NBC's "Today" program, will be named Friday as the new **president** of **NBC's entertainment division**, replacing Garth Ancier, NBC executives said.

Figure 1: A document from ACE 2004 with all entity mentions in bold.

Previous work on the use of kernels for RE has exploited some similarity measures over diverse features (Zelenko et al., 2002; Culotta and Sorensen, 2004; Zhang et al., 2005) or subsequence kernels over dependency graphs (Bunescu and Mooney, 2005a; Wang, 2008). More specifically, (Bunescu and Mooney, 2005a; Culotta and Sorensen, 2004) use kernels over dependency trees, which showed much lower accuracy than feature-based methods (Zhao and Grishman, 2005). One problem of the dependency kernels above is that they do not exploit the overall structural aspects of dependency trees. A more effective solution is the application of convolution kernels to constituent parse trees (Zhang et al., 2006) but this is not satisfactory from a general per-

This work has been partially funded by the LiveMemories project (http://www.livememories.org/) and Expert System (http://www.expertsystem.net/) research grant.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1378–1387,
Singapore, 6-7 August 2009.

spective since dependency structures offer some unique advantages, which should be exploited by an appropriate kernel.

Therefore, studying convolution tree kernels for dependency trees is worthwhile also considering that, to the best of our knowledge, these models have not been previously used for relation extraction[1] task. Additionally, sequence kernels should be included in such global study since some of their forms have not been applied to RE.

In this paper, we study and evaluate diverse convolution and sequence kernels for the RE problem by providing several kernel combinations on constituent and dependency trees and sequential structures. To fully exploit the potential of dependency trees, in addition to the SST kernel, we applied the partial tree (PT) kernel proposed in (Moschitti, 2006), which is a general convolution tree kernel adaptable for dependency structures. We also investigate various sequence kernels (e.g. the word sequence kernel (WSK) (Cancedda et al., 2003)) by incorporating dependency structures into word sequences. These are also enriched by including information from constituent parse trees.

We conduct experiments on the standard ACE 2004 newswire and broadcast news domain. The results show that although some kernels are less effective than others, they exhibit properties that are complementary to each other. In particular, we found that relation extraction can benefit from increasing the feature space by combining kernels (with a simple summation) exploiting the two different parsing paradigms. Our experiments on RE show that the current composite kernel, which is constituent-based is more effective than those based on dependency trees and individual sequence kernel but at the same time their combinations, i.e. dependency plus constituent trees, improve the state-of-the-art in RE. More interestingly, also the combinations of various sequence kernels gain significant better performance than the current state-of-the-art (Zhang et al., 2005).

Overall, these results are interesting for the computational linguistics research since they show that the above two parsing paradigms provide different and important information for a semantic task such as RE. Regarding sequence-based kernels, the WSK gains better performance than previous sequence and dependency models for RE.

[1]The function defined on (Culotta and Sorensen, 2004), although on dependency trees, is not a convolution tree kernel.

A review of previous work on RE is described in Section 2. Section 3 introduces support vector machines and kernel methods whereas our specific kernels for RE are described is Section 4. The experiments and conclusions are presented in sections 5 and 6, respectively.

2 Related Work

To identify semantic relations using machine learning, three learning settings have mainly been applied, namely supervised methods (Miller et al., 2000; Zelenko et al., 2002; Culotta and Sorensen, 2004; Kambhatla, 2004; Zhou et al., 2005), semi supervised methods (Brin, 1998; Agichtein and Gravano, 2000), and unsupervised method (Hasegawa et al., 2004). In a supervised learning setting, representative related work can be classified into generative models (Miller et al., 2000), feature-based (Roth and tau Yih, 2002; Kambhatla, 2004; Zhao and Grishman, 2005; Zhou et al., 2005) or kernel-based methods (Zelenko et al., 2002; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005a; Zhang et al., 2005; Wang, 2008; Zhang et al., 2006).

The learning model employed in (Miller et al., 2000) used statistical parsing techniques to learn syntactic parse trees. It demonstrated that a lexicalized, probabilistic context-free parser with head rules can be used effectively for information extraction. Meanwhile, feature-based approaches often employ various kinds of linguistic, syntactic or contextual information and integrate into the feature space. (Roth and tau Yih, 2002) applied a probabilistic approach to solve the problems of named entity and relation extraction with the incorporation of various features such as word, part-of-speech, and semantic information from WordNet. (Kambhatla, 2004) employed maximum entropy models with diverse features including words, entity and mention types and the number of words (if any) separating the two entities.

Recent work on Relation Extraction has mostly employed kernel-based approaches over syntactic parse trees. Kernels on parse trees were pioneered by (Collins and Duffy, 2001). This kernel function counts the number of common subtrees, weighted appropriately, as the measure of similarity between two parse trees. (Culotta and Sorensen, 2004) extended this work to calculate kernels between augmented dependency trees. (Zelenko et al., 2002) proposed extracting

relations by computing kernel functions between parse trees. (Bunescu and Mooney, 2005a) proposed a shortest path dependency kernel by stipulating that the information to model a relationship between two entities can be captured by the shortest path between them in the dependency graph.

Although approaches in RE have been dominated by kernel-based methods, until now, most of research in this line has used the kernel as some similarity measures over diverse features (Zelenko et al., 2002; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005a; Zhang et al., 2005; Wang, 2008). These are not convolution kernels and produce a much lower number of substructures than the PT kernel. A recent approach successfully employs a convolution tree kernel (of type SST) over constituent syntactic parse tree (Zhang et al., 2006; Zhou et al., 2007), but it does not capture grammatical relations in dependency structure. We believe that an efficient and appropriate kernel can be used to solve the RE problem, exploiting the advantages of dependency structures, convolution tree kernels and sequence kernels.

3 Support Vector Machines and Kernel Methods

In this section we give a brief introduction to support vector machines, kernel methods, diverse tree and sequence kernel spaces, which can be applied to the RE task.

3.1 Support Vector Machines (SVMs)

Support Vector Machines refer to a supervised machine learning technique based on the latest results of the statistical learning theory (Vapnik, 1998). Given a vector space and a set of training points, i.e. positive and negative examples, SVMs find a separating hyperplane $H(\vec{x}) = \vec{\omega} \times \vec{x} + b = 0$ where $\omega \in R^n$ and $b \in R$ are learned by applying the Structural Risk Minimization principle (Vapnik, 1995). SVMs is a binary classifier, but it can be easily extended to multi-class classifier, e.g. by means of the *one-vs-all* method (Rifkin and Poggio, 2002).

One strong point of SVMs is the possibility to apply kernel methods (robert Mller et al., 2001) to implicitly map data in a new space where the examples are *more easily* separable as described in the next section.

3.2 Kernel Methods

Kernel methods (Schlkopf and Smola, 2001) are an attractive alternative to feature-based methods since the applied learning algorithm only needs to compute a product between a pair of objects (by means of kernel functions), avoiding the explicit feature representation. A kernel function is a scalar product in a possibly unknown feature space. More precisely, The object o is mapped in $\vec{x}$ with a feature function $\phi : \mathcal{O} \rightarrow \Re^n$, where $\mathcal{O}$ is the set of the objects.

The kernel trick allows us to rewrite the decision hyperplane as:

$$H(\vec{x}) = \Big(\sum_{i=1..l} y_i \alpha_i \vec{x}_i \Big) \cdot \vec{x} + b =$$

$$\sum_{i=1..l} y_i \alpha_i \vec{x}_i \cdot \vec{x} + b = \sum_{i=1..l} y_i \alpha_i \phi(o_i) \cdot \phi(o) + b,$$

where y_i is equal to 1 for positive and -1 for negative examples, $\alpha_i \in \Re$ with $\alpha_i \geq 0$, $o_i \; \forall i \in \{1, .., l\}$ are the training instances and the product $K(o_i, o) = \langle \phi(o_i) \cdot \phi(o) \rangle$ is the kernel function associated with the mapping ϕ.

Kernel engineering can be carried out by combining basic kernels with additive or multiplicative operators or by designing specific data objects (vectors, sequences and tree structures) for the target tasks.

Regarding NLP applications, kernel methods have attracted much interest due to their ability of implicitly exploring huge amounts of structural features automatically extracted from the original object representation. The kernels for structured natural language data, such as parse tree kernel (Collins and Duffy, 2001) and string kernel (Lodhi et al., 2002) are examples of the well-known convolution kernels used in many NLP applications.

Tree kernels represent trees in terms of their substructures (called tree fragments). Such fragments form a feature space which, in turn, is mapped into a vector space. Tree kernels measure the similarity between pair of trees by counting the number of fragments in common. There are three important characterizations of fragment type (Moschitti, 2006): the SubTrees (ST), the SubSet Trees (SST) and the Partial Trees (PT). For sake of space, we do not report the mathematical description of them, which is available in (Vishwanathan and Smola, 2002), (Collins and Duffy,

2001) and (Moschitti, 2006), respectively. In contrast, we report some descriptions in terms of feature space that may be useful to understand the new engineered kernels.

In principle, a SubTree (ST) is defined by taking any node along with its descendants. A Sub-Set Tree (SST) is a more general structure which does not necessarily include all the descendants. It must be generated by applying the same grammatical rule set, which generated the original tree. A Partial Tree (PT) is a more general form of substructures obtained by relaxing constraints over the SST.

4 Kernels for Relation Extraction

In this section we describe the previous kernels based on constituent trees as well as new kernels based on diverse types of trees and sequences for relation extraction. As mentioned in the previous section, we can engineer kernels by combining tree and sequence kernels. Thus we focus on the problem to define structure embedding the desired syntactic relational information between two named entities (NEs).

4.1 Constituent and Dependency Structures

Syntactic parsing (or syntactic analysis) aims at identifying grammatical structures in a text. A parser thus captures the hidden hierarchy of the input text and processes it into a form suitable for further processing. There are two main paradigms for representing syntactic information: constituent and dependency parsing, which produces two different tree structures.

Constituent tree encodes structural properties of a sentence. The parse tree contains constituents, such as noun phrases (NP) and verb phrases (VP), as well as terminals/part-of-speech tags, such as determiners (DT) or nouns (NN). Figure 2.a shows the constituent tree of the sentence: *In Washington, U.S. officials are working overtime*.

Dependency tree encodes grammatical relations between words in a sentence with the words as nodes and dependency types as edges. An edge from a word to another represents a grammatical relation between these two. Every word in a dependency tree has exactly one parent except the root. Figure 2.b shows and example of the dependency tree of the previous sentence.

Given two NEs, such as *Washington* and *officials*, both the above trees can encode the syntactic dependencies between them. However, since each parse tree corresponds to a sentence, there may be more than two NEs and many relations expressed in a sentence. Thus, the use of the entire parse tree of the whole sentence holds two major drawbacks: first, it may be too computationally expensive for kernel calculation since the size of a complete parse tree may be very large (up to 300 nodes in the Penn Treebank (Marcus et al., 1993)); second, there is ambiguity on the target pairs of NEs, i.e. different NEs associated with different relations are described by the same parse tree. Therefore, it is necessary to identify the portion of the parse tree that best represent the useful syntactic information.

Let e_1 and e_2 be two entity mentions in the same sentence such that they are in a relationship R. For the constituent parse tree, we used the path-enclosed tree (PET), which was firstly proposed in (Moschitti, 2004) for Semantic Role Labeling and then adapted by (Zhang et al., 2005) for relation extraction. It is the smallest common subtree including the two entities of a relation. The dashed frame in Figure 2.a surrounds PET associated with the two mentions, *officials* and *Washington*. Moreover, to improve the representation, two extra nodes T1-PER, denoting the type PERSON, and T2-LOC, denoting the type LOCATION, are added to the parse tree, above the two target NEs, respectively. In this example, the above PET is designed to capture the relation *Located-in* between the entities "officials" and "Washington" from the ACE corpus. Note that, a third NE, *U.S.*, is characterized by the node GPE (GeoPolitical Entity), where the absence of the prefix T1 or T2 before the NE type (i.e. GPE), denotes that the NE does not take part in the target relation.

In previous work, some dependency trees have been used (Bunescu and Mooney, 2005a; Wang, 2008) but the employed kernel just exploited the syntactic information concentrated in the path between e_1 and e_2. In contrast, we defined and studied three different dependency structures whose potential can be fully exploited by our convolution partial tree kernel:

- Dependency Words (DW) tree is similar to PET adapted for dependency tree constituted by simple words. We select the minimal subtree which includes e_1 and e_2, and we insert an extra node as father of the NEs, labeled with the NE category. For example, given

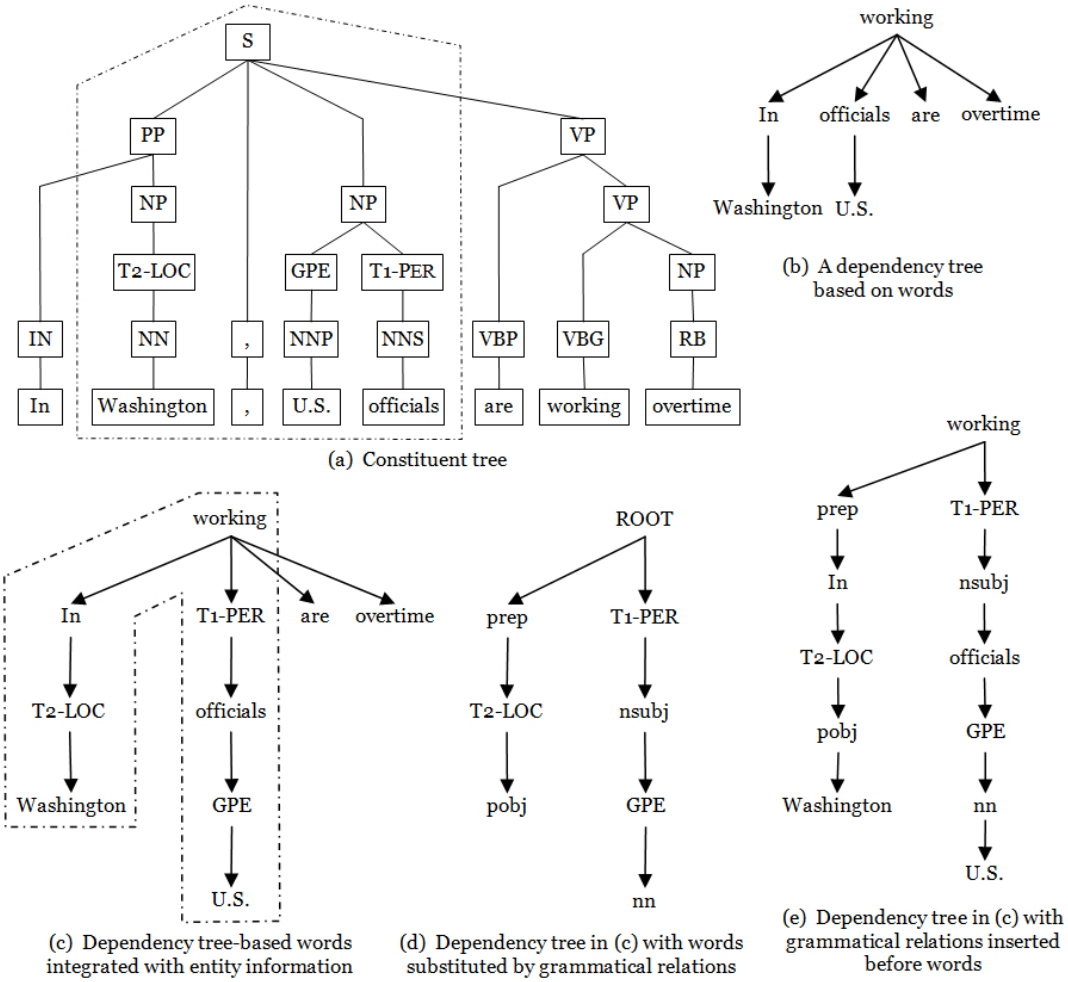


Figure 2: The constituent and dependency parse trees integrated with entity information

the tree in Figure 2.b, we design the tree in Figure 2.c surrounded by the dashed frames, where T1-PER, T2-LOC and GPE are the extra nodes inserted as fathers of *Washington*, *soldier* and *U.S.*.

- Grammatical Relation (GR) tree, i.e. the DW tree in which words are replaced by their grammatical functions, e.g. *prep*, *pobj* and *nsubj*. For example, Figure 2.d, shows the GR tree for the previous relation: *In* is replaced by *prep* , *U.S.* by *nsubj* and so on.

- Grammatical Relation and Words (GRW) tree, words and grammatical functions are both used in the tree, where the latter are inserted as a father node of the former. For example, Figure 2.e, shows such tree for the previous relation.

4.2 Sequential Structures

Some sequence kernels have been used on dependency structures (Bunescu and Mooney, 2005b; Wang, 2008). These kernels just used lexical words with some syntactic information. To fully exploit syntactic and semantic information, we defined and studied six different sequences (in a style similar to what proposed in (Moschitti, 2008)), which include features from constituent and dependency parse trees and NEs:

1. Sequence of terminals (lexical words) in the PET (SK_1), e.g.:
   T2-LOC Washington , U.S. T1-PER officials.

2. Sequence of part-of-speech (POS) tags in the PET (SK_2), i.e. the SK_1 in which words are replaced by their POS tags, e.g.:
   T2-LOC NN , NNP T1-PER NNS.

3. Sequence of grammatical relations in the

PET (SK_3), i.e. the SK_1 in which words are replaced by their grammatical functions, e.g.: *T2-LOC pobj , nn T1-PER nsubj*.

4. Sequence of words in the DW (SK_4), e.g.: *Washington T2-LOC In working T1-PER officials GPE U.S.*.

5. Sequence of grammatical relations in the GR (SK_5), i.e. the SK_4 in which words are replaced by their grammatical functions, e.g.: *pobj T2-LOC prep ROOT T1-PER nsubj GPE nn*.

6. Sequence of POS tags in the DW (SK_6), i.e. the SK_4 in which words are replaced by their POS tags, e.g.: *NN T2-LOC IN VBP T1-PER NNS GPE NNP*.

It is worth noting that the potential information contained in such sequences can be fully exploited by the word sequence kernel.

4.3 Combining Kernels

Given that syntactic information from different parse trees may have different impact on relation extraction (RE), the viable approach to study the role of dependency and constituent parsing is to experiment with different syntactic models and measuring the impact in terms of RE accuracy. For this purpose we compared the composite kernel described in (Zhang et al., 2006) with the partial tree kernels applied to DW, GR, and GRW and sequence kernels based on six sequences described above. The composite kernels include polynomial kernel applied to entity-related feature vector. The word sequence kernel (WSK) is always applied to sequential structures. The used kernels are described in more detail below.

4.3.1 Polynomial Kernel

The basic kernel between two named entities of the ACE documents is defined as:

$$K_P(R_1, R_2) = \sum_{i=1,2} K_E(R_1.E_i, R_2.E_i),$$

where R_1 and R_2 are two relation instances, E_i is the i^{th} entity of a relation instance. $K_E(\cdot, \cdot)$ is a kernel over entity features, i.e.:

$$K_E(E_1, E_2) = (1 + \vec{x}_1 \cdot \vec{x}_2)^2,$$

where $\vec{x}_1$ and $\vec{x}_2$ are two feature vectors extracted from the two NEs.

For the ACE 2004, the features used include: entity headword, entity type, entity subtype, mention type, and LDC[2] mention type. The last four attributes are taken from the ACE corpus 2004. In ACE, each mention has a head annotation and an extent annotation.

4.3.2 Kernel Combinations

1. Polynomial kernel plus a tree kernel:

$$CK_1 = \alpha \cdot K_P + (1 - \alpha) \cdot K_x,$$

where α is a coefficient to give more impact to K_P and K_x is either the partial tree kernel applied to one the possible dependency structures, DW, GR or GRW or the SST kernel applied to PET, described in the previous section.

2. Polynomial kernel plus constituent plus dependency tree kernels:

$$CK_2 = \alpha \cdot K_P + (1 - \alpha) \cdot (K_{SST} + K_{PT})$$

where K_{SST} is the SST kernel and K_{PT} is the partial tree kernel (applied to the related structures as in point 1).

3. Constituent tree plus square of polynomial kernel and dependency tree kernel:

$$CK_3 = \alpha \cdot K_{SST} + (1 - \alpha) \cdot (K_P + K_{PT})^2$$

4. Dependency word tree plus grammatical relation tree kernels:

$$CK_4 = K_{PT-DW} + K_{PT-GR}$$

where K_{PT-DW} and K_{PT-GR} are the partial tree kernels applied to dependency structures DW and GR.

5. Polynomial kernel plus dependency word plus grammatical relation tree kernels:

$$CK_5 = \alpha \cdot K_P + (1-\alpha) \cdot (K_{PT-DW} + K_{PT-GR})$$

Some preliminary experiments on a validation set showed that the second, the fourth and the fifth combinations yield the best performance with $\alpha = 0.4$ while the first and the third combinations yield the best performance with $\alpha = 0.23$.

Regarding WSK, the following combinations are applied:

[2]Linguistic Data Consortium (LDC): http://www.ldc.upenn.edu/Projects/ACE/

1. $SK_3 + SK_4$
2. $SK_3 + SK_6$
3. $SSK = \sum_{i=1,..,6} SK_i$
4. $K_{SST} + SSK$
5. $CSK = \alpha \cdot K_P + (1 - \alpha) \cdot (K_{SST} + SSK)$

Preliminary experiments showed that the last combination yields the best performance with $\alpha = 0.23$.

We used a polynomial expansion to explore the bi-gram features of i) the first and the second entity participating in the relation, ii) grammatical relations which replace words in the dependency tree. Since the kernel function set is closed under normalization, polynomial expansion and linear combination (Schlkopf and Smola, 2001), all the illustrated composite kernels are also proper kernels.

5 Experiments

Our experiments aim at investigating the effectiveness of convolution kernels adapted to syntactic parse trees and various sequence kernels for the RE task. For this purpose, we use the subset and partial tree kernel over different kinds of trees, namely constituent and dependency syntactic parse trees. Diverse sequences are applied individually and in combination together. We consider our task of relation extraction as a classification problem where categories are relation types. All pairs of entity mentions in the same sentence are taken to generate potential relations, which will be processed as positive and negative examples.

5.1 Experimental setup

We use the newswire and broadcast news domain in the English portion of the ACE 2004 corpus provided by LDC. This data portion includes 348 documents and 4400 relation instances. It defines seven entity types and seven relation types. Every relation is assigned one of the seven types: Physical, Person/Social, Employment/Membership/-Subsidiary, Agent-Artifact, PER/ORG Affiliation, GPE Affiliation, and Discourse. For sake of space, we do not explain these relationships here, nevertheless, they are explicitly described in the ACE document guidelines. There are 4400 positive and 38,696 negative examples when generating pairs of entity mentions as potential relations.

Documents are parsed using Stanford Parser (Klein and Manning, 2003) to produce parse trees. Potential relations are generated by iterating all pairs of entity mentions in the same sentence. Entity information, namely entity type, is integrated into parse trees. To train and test our binary relation classifier, we used SVMs. Here, relation detection is formulated as a multiclass classification problem. The *one vs. rest* strategy is employed by selecting the instance with largest margin as the final answer. For experimentation, we use 5-fold cross-validation with the Tree Kernel Tools (Moschitti, 2004) (available at http://disi.unitn.it/~moschitt/Tree-Kernel.htm).

5.2 Results

In this section, we report the results of different kernels setup over constituent (CT) and dependency (DP) parse trees and sequences taken from these parse trees. The tree kernel (TK), composite kernel (CK_1, CK_2, CK_3, CK_4, and CK_5 corresponding to five combination types in Section 4.3.2) were employed over these two syntactic trees. For the tree kernel, we apply the SST kernel for the path-enclosed tree (PET) of the constituent tree and the PT kernel for three kinds of dependency tree DW, GR, and GRW, described in the previous section. The two composite kernels CK_2 and CK_3 are applied over both two parse trees. The word sequence kernels are applied over six sequences $SK_1, SK_2, SK_3, SK_4, SK_5$, and SK_6 (described in Section 4.3).

The results are shown in Table 1 and Table 2. In the first table, the first column indicates the structure used in the combination shown in the second column, e.g. PET associated with CK_1 means that the SST kernel is applied on PET (a portion of the constituent tree) and combined with the CK_1 schema whereas PET and GR associated with CK_5 means that SST kernel is applied to PET and PT kernel is applied to GR in CK_5. The remaining three columns report Precision, Recall and F1 measure. The interpretation of the second table is more immediate since the only tree kernel involved is the SST kernel applied to PET and combined by means of CK_1.

We note that: first, the dependency kernels, i.e. the results on the rows from 3 to 6 are below the composite kernel CK_1, i.e. 68.9. This is the state-of-the-art in RE, designed by (Zhang et al., 2006), where our implementation provides

Parse Tree	Kernel	P	R	F
PET	$\mathbf{CK_1}$	**69.5**	**68.3**	**68.9**
DW	CK_1	53.2	59.7	56.3
GR	CK_1	58.8	61.7	60.2
GRW	CK_1	56.1	61.2	58.5
DW and GR	CK_5	59.7	64.1	61.8
PET and GR	$\mathbf{CK_2}$	**70.7**	**69.0**	**69.8**
	$\mathbf{CK_3}$	**70.8**	**70.2**	**70.5**

Table 1: Results on the ACE 2004 evaluation test set. Six structures were experimented over the constituent and dependency trees.

Kernel	P	R	F
$\mathbf{CK_1}$	**69.5**	**68.3**	**68.9**
SK_1	72.0	52.8	61.0
SK_2	61.7	60.0	60.8
SK_3	62.6	60.7	61.6
SK_4	73.1	50.3	59.7
SK_5	59.0	60.7	59.8
SK_6	57.7	61.8	59.7
$\mathbf{SK_3 + SK_4}$	**75.0**	**63.4**	**68.8**
$SK_3 + SK_6$	66.8	65.1	65.9
$\mathbf{SSK = \sum_i SK_i}$	**73.8**	**66.2**	**69.8**
CSK	**75.6**	**66.6**	**70.8**
$\mathbf{CK_1 + SSK}$	**76.6**	**67.0**	**71.5**
(Zhou et al., 2007) CK_1 *with Heuristics*	*82.2*	*70.2*	*75.8*

Table 2: Performance comparison on the ACE 2004 data with different kernel setups.

a slightly smaller result than the original version (i.e. an F1 of about 72 using a different syntactic parser).

Second, CK_1 improves to 70.5, when the contribution of PT kernel applied to GR (dependency tree built using grammatical relations) is added. This suggests that dependency structures are effectively exploited by PT kernel and that such information is somewhat complementary to constituent trees.

Third, in the second table, the model $CK_1 + SSK$, which adds to CK_1 the contribution of diverse sequence kernels, outperforms the state-of-the-art by 2.6%. This suggests that the sequential information encoded by several sequence kernels can better represents the dependency information.

Finally, we also report in the last row (in italic) the superior RE result by (Zhou et al., 2007). However, to achieve this outcome the authors used the composite kernel CK_1 with several heuristics to define an effective portion of constituent trees. Such heuristics expand the tree and remove unnecessary information allowing a higher improvement on RE. They are tuned on the target RE task so although the result is impressive, we cannot use it to compare with pure automatic learning approaches, such us our models.

6 Conclusion and Future Work

In this paper, we study the use of several types of syntactic information: constituent and dependency syntactic parse trees. A relation is represented by taking the path-enclosed tree (PET) of the constituent tree or of the path linking two entities of the dependency tree. For the design of automatic relation classifiers, we have investigated the impact of dependency structures to the RE task. Our novel composite kernels, which account for the two syntactic structures, are experimented with the appropriate convolution kernels and show significant improvement with respect to the state-of-the-art in RE.

Regarding future work, there are many research line that may be followed:

i) Capturing more features by employing external knowledge such as ontological, lexical resource or WordNet-based features (Basili et al., 2005a; Basili et al., 2005b; Bloehdorn et al., 2006; Bloehdorn and Moschitti, 2007) or shallow semantic trees, (Giuglea and Moschitti, 2004; Giuglea and Moschitti, 2006; Moschitti and Bejan, 2004; Moschitti et al., 2007; Moschitti, 2008; Moschitti et al., 2008).

ii) Design a new tree-based structures, which combines the information of both constituent and dependency parses. From dependency trees we can extract more precise but also more sparse relationships (which may cause overfit). From constituent trees, we can extract subtrees constituted by non-terminal symbols (grammar symbols), which provide a better generalization (with a risk of underfitting).

iii) Design a new kernel which can integrate the advantages of the constituent and dependency tree. The new tree kernel should inherit the benefits of the three available tree kernels: ST, SST or PT.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text col-

lections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*.

Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2005a. Effective use of WordNet semantics via kernel-based learning. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 1–8, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2005b. A semantic kernel to classify texts with very few training examples. In *In Proceedings of the Workshop on Learning in Web Search, at the*.

Stephan Bloehdorn and Alessandro Moschitti. 2007. Structure and semantics for expressive text kernels. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 861–864, New York, NY, USA. ACM.

Stephan Bloehdorn, Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2006. Semantic kernels for text classification based on topological measures of feature similarity. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 06), Hong Kong, 18-22 December 2006*, DEC.

Sergey Brin. 1998. Extracting patterns and relations from world wide web. In *Proceeding of WebDB Workshop at 6th International Conference on Extending Database Technology*, pages 172–183.

Razvan C. Bunescu and Raymond J. Mooney. 2005a. A shortest path dependency kernel for relation extraction. In *Proceedings of EMNLP*, pages 724–731.

Razvan C. Bunescu and Raymond J. Mooney. 2005b. Subsequence kernels for relation extraction. In *Proceedings of EMNLP*.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. 2003. Word sequence kernels. *Journal of Machine Learning Research*, pages 1059–1082.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Proceedings of Neural Information Processing Systems (NIPS'2001)*.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on ACL*, Barcelona, Spain.

Ana-Maria Giuglea and Alessandro Moschitti. 2004. Knowledge discovery using framenet, verbnet and propbank. In A. Meyers, editor, *Workshop on Ontology and Knowledge Discovering at ECML 2004*, Pisa, Italy.

Ana-Maria Giuglea and Alessandro Moschitti. 2006. Semantic Role Labeling via Framenet, Verbnet and Propbank. In *Proceedings of ACL 2006*, Sydney, Australia.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting on ACL*, Barcelona, Spain.

Nanda Kambhatla. 2004. Combining lexical, syntactic and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, Barcelona, Spain.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the ACL*, pages 423–430.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, , and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, pages 419–444.

Mitchell P. Marcus, Beatrice Santorini, , and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

Scott Miller, Heidi Fox, Lance Ramshaw, , and Ralph Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proceedings of the 1st conference on North American chapter of the ACL*, pages 226–233, Seattle, USA.

Alessandro Moschitti and Cosmin Bejan. 2004. A semantic kernel for predicate argument classification. In *CoNLL-2004*, Boston, MA, USA.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proceedings of ACL'07*, Prague, Czech Republic.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of the 42nd Meeting of the ACL*, Barcelona, Spain.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of the 17th European Conference on Machine Learning*, Berlin, Germany.

Alessandro Moschitti. 2008. Kernel methods, syntax and semantics for relational text categorization. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 253–262, New York, NY, USA. ACM.

Ryan Michael Rifkin and Tomaso Poggio. 2002. *Everything old is new again: a fresh look at historical approaches in machine learning*. PhD thesis, Massachusetts Institute of Technology.

Klaus robert Mller, Sebastian Mika, Gunnar Rtsch, Koji Tsuda, , and Bernhard Schlkopf. 2001. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201.

Dan Roth and Wen tau Yih. 2002. Probabilistic reasoning for entity and relation recognition. In *Proceedings of the COLING-2002*, Taipei, Taiwan.

Bernhard Schlkopf and Alexander J. Smola. 2001. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer–Verlag, New York.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley and Sons, New York.

S.V.N. Vishwanathan and Alexander J. Smola. 2002. Fast kernels on strings and trees. In *Proceedings of Neural Information Processing Systems*.

Mengqiu Wang. 2008. A re-examination of dependency path kernels for relation extraction. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing-IJCNLP*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. In *Proceedings of EMNLP-ACL*, pages 181–201.

Min Zhang, Jian Su, Danmei Wang, Guodong Zhou, and Chew Lim Tan. 2005. Discovering relations between named entities from a large raw corpus using tree similarity-based clustering. In *Proceedings of IJCNLP'2005, Lecture Notes in Computer Science (LNCS 3651)*, pages 378–389, Jeju Island, South Korea.

Min Zhang, Jie Zhang, Jian Su, , and Guodong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of COLING-ACL 2006*, pages 825–832.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Meeting of the ACL*, pages 419–426, Ann Arbor, Michigan, USA.

GuoDong Zhou, Jian Su, Jie Zhang, , and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Meeting of the ACL*, pages 427–434, Ann Arbor, USA, June.

GuoDong Zhou, Min Zhang, DongHong Ji, and QiaoMing Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of EMNLP-CoNLL 2007*, pages 728–736.

Automatic Acquisition of the *Argument-Predicate* Relations from a Frame-Annotated Corpus

Ekaterina Ovchinnikova
University of Osnabrück
eovchinn@uos.de

Theodore Alexandrov
University of Bremen
theodore@math.uni-bremen.de

Tonio Wandmacher
University of Osnabrück
twandmac@uos.de

Abstract

This paper presents an approach to automatic acquisition of the argument-predicate relations from a semantically annotated corpus. We use SALSA, a German newspaper corpus manually annotated with role-semantic information based on frame semantics. Since the relatively small size of SALSA does not allow to estimate the semantic relatedness in the extracted argument-predicate pairs, we use a larger corpus for ranking. Two experiments have been performed in order to evaluate the proposed approach. In the first experiment we compare automatically extracted argument-predicate relations with the gold standard formed from associations provided by human subjects. In the second experiment we calculate correlation between automatic relatedness measure and human ranking of the extracted relations.

1 Introduction

There are many debates in lexical semantics about what kind of world knowledge actually belongs to the meaning of a lexeme. Nowadays, it is widely accepted that predicates impose selectional restrictions on their arguments. For example, since we know that the predicate *to be hungry* mainly takes expressions describing animate beings as arguments, we can correctly resolve the anaphora in the following sentence: *We gave the bananas to the monkeys because they were hungry*. There exists also multiple linguistic evidence showing that the semantics of arguments can help to predict implicit predicates. For example, the sentence *John finished the cigarette* usually means *John finished smoking the cigarette* because the meaning of the noun *cigarette* is strongly associated with the smoking activity.

It has been claimed that information about predicates associated with nouns can be helpful for a variety of tasks in natural language processing (NLP), see for example (Pustejovsky et al., 1993; Voorhees, 1994). However, at present there exists no corresponding lexical semantic resource. Several approaches have been presented that aim at creating a knowledge base containing noun-verb relations. There are two main research paradigms for developing such knowledge bases. The first paradigm assumes manual development of the resource (Pustejovsky et al., 2006), while the second one relies on automatic acquisition methods, see for example (Cimiano and Wenderoth, 2007). In this paper we propose a procedure for automatic acquisition of argument-predicate relations from a semantically annotated corpus. In line with (Lapata and Lascarides, 2003) our approach is based on the assumption that predicates are omitted in a discourse when they are highly predictable from the semantics of their arguments. We exploit SALSA (Burchardt et al., 2006), a German newspaper corpus manually annotated with FrameNet frames based on frame semantics. Using a manually annotated corpus for relation extraction has one particular advantage compared to extraction from plain text: the type of an argument-predicate relation is already annotated; there is no need to determine it by automatic means which are usually error-prone. However, the relatively small size of SALSA does not allow to make relevant predictions about the degree of semantic relatedness in the extracted argument-predicate pairs, see section 4. We therefore employ a considerably larger unannotated corpus for weighting. The results are evaluated quantitatively against human judgments obtained experimentally. The proposed evaluation procedure is similar to that presented in (Cimiano and Wenderoth, 2007). First, we create a gold standard for 30 words from the argument list and evaluate our approach with respect to this

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1388–1397,
Singapore, 6-7 August 2009.

gold standard. Second, we provide results from an evaluation in which test subjects are asked to rate automatically extracted relations using a four-point scale.

The paper is structured as follows: Section 2 describes some linguistic phenomena requiring inferences of an implicit predicate from the semantics of an explicitly given argument. In section 3 we give a short overview of the related work. Sections 4 discusses the SALSA corpus. Section 5 introduces our approach. Finally, section 6 describes an experimental evaluation of the presented approach and section 7 concludes the paper.

2 Implicit Predicates

In this section we discuss some linguistic phenomena requiring inferences of an implicit predicate from the semantics of an explicitly given argument for their resolution. One of the most studied phenomena that Pustejovsky (1991) has called logical metonymy is illustrated by the examples (1a) and (1b) below. In the case of logical metonymy an implicit predicate is inferable from particular verb-noun and adjective-noun pairs in a systematic way. The verb *anfangen* 'to start' and the adjective *kompliziert* 'complicated' in the mentioned examples semantically select for an event, while the nouns (*Buch* 'book' and *Frage* 'question' respectively) have a different semantic type. However, the set of the most probable implicit predicates is predictable from the semantics of the nouns. Thus, (1a) plausibly means *Als ich angefangen habe, dieses Buch zu lesen/schreiben...* 'When I have started to read/write this book...' and (2a) plausibly means *eine Frage die kompliziert zu beantworten ist* 'a question which is complicated to answer'.

Example 1

(a) *Als ich mit diesem Buch angefangen habe...*
'When I have started this book...'
(b) *eine komplizierte Frage*
'a complicated question'
(c) *Studentenfutter*
'student food'
(d) *Nachrichtenagentur Xinhua über Beziehungen beider Seiten der Taiwan-Strasse*
'News agency Xinhua about relations of both sides of the Taiwan Strait'
(e) *Hans ist beredt*
'Hans is eloquent'

As we can see from Example 1, besides logical metonymy there are other linguistic phenomena requiring knowledge about predicates associated with an argument for their resolution. Example (1c) contains a noun compound which can be interpreted on basis of the meaning of the noun *Futter* 'food'. In general, noun compounds can be interpreted in many different ways depending on the semantics of the constituencies: *morning coffee* is a coffee which is drunk in the morning, *brick house* is a house which is made of bricks etc. In case of (1c) the relation via the predicate *essen* 'to eat' taking *Studenten* 'students' as a subject and *Futter* 'food' as an object seems to be the most plausible one.

The phrase (1d) is a title of a newspaper article. As in the previous examples, a predicate is left out in (1d). The meaning of the preposition *über* 'about' can help to narrow down the set of possible predicates, but still allows an inadequately large range of interpretations. However, the semantics of the noun *Nachrichtenagentur* 'news agency' supports such interpretations as *berichten* 'to report' or *informieren* 'to inform'.

Most of the literature discusses predicates inferable from nouns. However, other parts of speech can support similar inferences. In example (1e) a predicate is predictable on the basis of the meaning of the adjective *beredt* 'eloquent'. The sentence (1e) most plausibly means that Hans speaks eloquently.

Example 1 shows that knowledge about predicates associated with explicitly given arguments can help to deal with several linguistic phenomena. The cases when a predictable predicate is left out are not rare in natural language. For example, for logical metonymy a corpus study has shown that the constructions like *begin V NP* occur rarely if the verb *V* corresponds to a highly plausible interpretation of *begin NP* (Briscoe et al., 1990).

3 Related Work

The most influential account of logical metonymy is provided by Pustejovsky's theory of the Generative Lexicon, GL (Pustejovsky, 1991). According to Pustejovsky the meaning of a noun includes a *qualia structure* representing "the essential attributes of an object as defined by the lexical item". Thus, the lexical meaning of the noun *book* includes *read* and *write* as qualia roles. In the framework of GL, Pustejovsky et al. (2006)

are manually developing the Brandeis Semantic Ontology which is a large generative lexicon ontology and dictionary. There also exist several approaches to automatic acquisition of qualia structures from text corpora which aim at supporting the time-consuming manual work. For example, Pustejovsky et al. (1993) use generalized syntactic patterns for extracting qualia structures from a partially parsed corpus. Cimiano and Wenderoth (2007) suggest a pattern-based method for automatic extraction of qualia structures from the Web. The results of the human judgment experiment reported in (Cimiano and Wenderoth, 2007) suggest that the automatic acquisition of qualia structures is a difficult task. Human test subjects have shown a very low agreement (11,8% average agreement) in providing qualia structures for given nouns.

Another line of research on inferring implicit predicates concerns using information about collocations derived from corpora. For example, Lapata and Lascarides (2003) resolve logical metonymy on the basis of the distribution of paraphrases like *finish the cigarette – finish smoking the cigarette* and *easy problem – problem which is easy to solve* in a corpus. This approach shows promising results, but it is limited to logical metonymy. Similarly, Nastase et al. (2006) use grammatical collocations for defining semantic relations between constituents in noun compounds.

In our study we aim at extracting intuitively plausible argument-predicate relations from a semantically annotated corpus. Using an annotated corpus we avoid problems of defining types of these relations by automatic means which are usually error-prone. We represent argument-predicate relations in terms of FrameNet frames which allow for a fine-grained and grounded representation supporting paraphrasing, see next sections. Our approach is not restricted to nouns. We also concern relations where argument positions are filled by adjectives, adverbs or even verbs.

4 The SALSA Corpus

For relation extraction we have chosen the SALSA corpus (Burchardt et al., 2006) developed at Saarland University. SALSA is a German corpus manually annotated with role-semantic information, based on the syntactically annotated TIGER newspaper corpus (Brants et al., 2002). The 2006 SALSA release which we have used contains about 20 000 annotated predicate instances. The corpus is annotated with the set of FrameNet frames.

The FrameNet, FN (Ruppenhofer et al., 2006), lexical resource is based on frame semantics (Fillmore, 1976), see *http://framenet.icsi.berkeley.edu*. The lexical meaning of predicates in FN is expressed in terms of frames (approx. 800 frames) which are supposed to describe prototypical situations spoken about in natural language. Every frame contains a set of roles (or frame elements, FEs) corresponding to the participants of the described situation. Predicates with similar semantics are assigned to the same frame, e.g. *to give* and *to hand over* refer to the GIVING frame. Consider a FN annotation for the sentence (2a) below. In this annotation DONOR, RECIPIENT and THEME are roles in the frame GIVING and *John*, *Mary* and *a book* are fillers of these roles. The FN annotation generalizes across near meaning-preserving transformations, see (2b).

Example 2

(a) [*John*]$_{\text{DONOR}}$ [*gave*]$_{\text{GIVING}}$ [*Mary*]$_{\text{RECIPIENT}}$ [*a book*]$_{\text{THEME}}$.
(b) [*John*]$_{\text{DONOR}}$ [*gave*]$_{\text{GIVING}}$ [*a book*]$_{\text{THEME}}$ [*to Mary*]$_{\text{RECIPIENT}}$.

In FN information about syntactic realization patterns of frame elements as well as information about frequency of occurrences of these patterns in corpora is provided. For example, the role DONOR in the frame GIVING is most frequently filled by a noun phrase in the subject position or by a prepositional phrase with the preposition *by* as the head in the complement position.

The FN project originally aimed at developing a frame-semantic lexicon for English. Later on FN frames turned out to be to a large extent language independent (Burchardt et al., 2006). In most of the cases German predicates could be successfully described by the FN frames. However, some of the frames required adaptation to the German data, e.g. new FEs were introduced. Since FN does not cover all possible word senses, new frames needed to be added for some of the predicates.

We have chosen the SALSA corpus for our experiments because to our knowledge it is the only freely available corpus which contains both syntactic and role-semantic annotation. However, we are aware that SALSA (approx. 700 000 tokens) is too small to compute a reliable co-occurrence model for measuring plausibility of the extracted argument-predicate relations, though it

is relatively large for a manually annotated corpus. As it was shown in (Bullinaria and Levy, 2007), co-occurrence-based approaches need very large training corpora in order to reliably compute semantic relatedness. The SALSA corpus, comprising less than 1 million tokens, is too small for this purpose. Moreover, a considerable number of predicates in SALSA appeared to be unannotated. Some of the high frequency pairs, as for example *Bombe, explodieren* 'bomb, to explode', occur in SALSA only once, just as occasional pairs like *Deutsche, entdecken* 'German, to discover'. We have tried to overcome the size problems by using a larger unannotated corpus for recomputing the rating of our resulting relations, see next section.

5 Automatic Acquisition of the *Argument-Predicate* Relations

In line with (Lapata and Lascarides, 2003), our approach to extraction of argument-predicate (AP) relations is based on two assumptions:

A1: If predicates are highly predictable from the semantics of their arguments then they can be omitted in a discourse;
A2: If a predicate frequently takes a word as an argument then it is highly predictable from the semantics of this word.

In the proposed experimental setting argument-predicate relations are defined in terms of the FrameNet frames. Thus, we aim at extracting from SALSA tuples of the form ⟨*Argument*, ROLE, FRAME, *Predicate*⟩ such that the *Argument* plausibly fills the ROLE in the FRAME evoked by the *Predicate*. As already mentioned in section 3, our approach is not restricted to nouns. We also treat arguments expressed by other content parts of speech. The proposed relation extraction procedure consists in

- finding for every content word which occurs in the corpus a set of predicates taking this word as an argument with a high probability;
- defining a relation between the word and every predicate from this set by finding which roles the noun fills in frames evoked by the predicate;
- estimating the degree of the semantic relatedness in the extracted argument-predicate pairs.

For example, analyzing the following sentence

[*Fünf Oppositionelle*]$_{\text{SUSPECT}}$ *sind in Ebebiyin* [*von der Polizei*]$_{\text{AUTHORITIES}}$ [*festgenommen*]$_{\text{ARREST}}$ *worden*.

'Five members of the opposition have been arrested by the police in Ebebiyin.'

we aim at extracting the following tuples:

Argument	Role	Frame	Predicate
Oppositionell	SUSPECT	ARREST	*festnehmen*
Polizei	AUTHORITIES	ARREST	*festnehmen*

Relation Extraction

In SALSA, every sentence is annotated with a set of frames in such a way that for every frame its FEs refer to some syntactic constituents in the sentence. In order to extract argument-predicate relations from SALSA we need 1) to find a content head for every constituent corresponding to a FE; 2) to resolve possibly existing anaphora. Since SALSA is syntactically annotated, the first task proved to be relatively easy.[1] On the contrary, anaphora resolution is well-known to be one of most challenging NLP tasks. In our study, we do not focus on it, and we treat only pronominal anaphora using the following straightforward resolution algorithm: given a pronoun the first noun which agrees in number and gender with the pronoun is supposed to be its antecedent. In order to evaluate this resolution procedure we have inspected 100 anaphoric cases. In approximately three fourths of the cases the anaphora were resolved correctly. Therefore, we have assigned a confidence rate of 0,75 to the FE fillers resulting from a resolved anaphora. In non-anaphoric cases a confidence rate of 1 was assigned.

For every extracted tuple of the form ⟨*Argument*, ROLE, FRAME, *Predicate*⟩ we have summed up the corresponding confidence rates. Finally, we have obtained around 30 000 tuples with confidence rates ranging from 0,75 to 88. It is not surprising that most of the arguments appeared to be nouns, while most of the predicates are expressed by verbs. Since SALSA has been annotated manually, there are almost no mistakes in defining types of the semantic

[1] We have excluded from the consideration foreign-language expressions, while proper nouns were treated in the usual way. For verb phrases with auxiliary or modal verbs as heads the main verb was taken as a corresponding role filler.

relations between arguments and predicates.[2] For several pairs, the semantic relation between an argument and a predicate is ambiguous. Consider the tuples extracted for the word pair *Buch, schreiben* 'book, to write' which are given below. While the first tuple corresponds to phrases like *ein Buch schreiben* 'to write a book', the second one abstracts from the expressions like *in einem Buch schreiben* 'to write in a book'.

Argument	Role	Frame	Predicate
Buch	TEXT	TEXT_CREATION	*schreiben*
Buch	MEDIUM	STATEMENT	*schreiben*

Additionally, ambiguity can arise because of the annotation disagreements in SALSA. For example, the pair (*Haft, sitzen*) 'imprisonment', 'to sit' in Table 1 was annotated in SALSA both with the BEING_LOCATED and with the POSTURE frames.

As mentioned in section 4, a considerable number of predicates in SALSA is not annotated semantically. In order to find out how many relevant AP-relations get lost if we consider only semantically annotated predicates, we have additionally extracted AP-pairs on the basis of the syntactic annotation only. The anaphora resolution procedure as described above was again applied to the syntactic argument heads. We have obtained around 56 500 pairs with confidence rates ranging from 0,75 to 71,50.[3]

As one could expect, being a newspaper corpus SALSA appeared to be thematically unbalanced. The most frequent argument-predicate relations occurring in SALSA reflect common topics discussed in newspapers: economics (e.g. (*Prozent, steigen*), 'percent', 'to increase'), criminality (e.g. (*Haft, verurteilen*) 'imprisonment', 'to sentence'), catastrophes (e.g. (*Mensch, töten*) 'human', 'to kill') etc.

Ranking

As mentioned in section 4, the size of SALSA does not allow to make relevant predictions about the distribution of frames and role fillers. Only 2% of the relations occur in SALSA more then 3 times. In order to overcome this problem we have developed a measure of semantic relatedness between the extracted arguments and predicates which takes into account their co-occurrence in a larger and more representative corpus. For computing semantic relatedness we have used a lemmatized newspaper corpus (Süddeutsche Zeitung, SZ) of 145 million words. Given a tuple t with a confidence rate c containing an argument a and a predicate p, the relatedness measure *rm* of t was computed as follows:

$$rm(t) = lsa(a, p) + c/max(c),$$

where the $lsa(a, p)$ is based on Latent Semantic Analysis, LSA (Deerwester et al., 1990). LSA is a vector-based technique that has been shown to give reliable estimates on semantic relatedness. It makes use of distributional similarities of words in text and constructs a semantic space (or word space) in which every word of a given vocabulary is represented as a vector. Such vectors can then be compared to one another by the usual vector similarity measures (e.g. cosine). We calculated the LSA word space using the Infomap toolkit10 v. 0.8.6 (*http://infomap-nlp.sourceforge.net*). The co-occurrence matrix (window size: 5 words) comprised 80 000×3 000 terms and was reduced by SVD to 300 dimensions. For the vector comparisons the cosine measure was applied. To those words which did not occur in the analyzed SZ corpus (approx. 3500 words) a *lsa* measure of 0 was assigned. To provide a comparable contribution to rm, the confidence rates c extracted from SALSA are divided by the maximal confidence rate. The *rm* function is a linear interpolation of the *lsa* and the normalized *c* measure. As mentioned above, the *c* measure is a discriminative factor for only 2% of the relations. For the remaining 98% the normalized *c* values are small (0,003 or 0,002 or 0,001). Therefore, calculating the *rm* measure we mainly rely on *lsa*, while normalized *c* actually plays a role only for the relations frequently occurring in SALSA. Table 1 contains the 5 most semantically related predicates for an example argument.

6 Evaluation

Since the extracted argument-predicate relations are intended to be used for inferring intuitively obvious predicates,we evaluate to which extend they correspond to human intuition.

[2] Mistakes can arise only because of the annotation errors and errors in the anaphora resolution procedure.

[3] The comparison of the results obtained by the extraction procedure based on the semantic annotation with the results of the procedure based on the syntactic annotation only is provided in the next section.

Table 1: Examples of the extracted argument-predicate relations

Argument	Role	Frame	Predicate	*rm*
Haft	FINDING	VERDICT	*verurteilen* 'to sentence'	0,939
'imprisonment'	LOCATION	BEING_LOCATED	*sitzen* 'to sit'	0,237
	LOCATION	POSTURE	*sitzen* 'to sit'	0,226
	MESSAGE	REQUEST	*fordern* 'to demand'	0,153
	BAD_OUTCOME	RUN_RISK-FNSALSA	*drohen* 'to threaten'	0,144

Gold Standard

Similar to (Cimiano and Wenderoth, 2007) we provide a gold standard for 30 test arguments occurring in the SALSA corpus. The test arguments were selected randomly from the set of those arguments that have more than one predicate associated with them such that a value of argument-predicate relatedness exceeds the average one. These words were nearly uniformly distributed among 20 participants of the experiment, who were all non-linguists. We also ensured that each word was treated by three different subjects. For every word we asked our subjects to write between 5 and 10 short phrases that contain a predicate taking the given word as an argument, e.g. *book – to read a book*. The participants were asked to provide phrases instead of single predicates, because we wanted to control the syntactic and semantic position of the arguments. The participants received an instruction informally describing the notion of predicate and what kind of phrases they are supposed to come up with. Besides the task description they were shown examples containing appropriate and inappropriate phrases. Some of the examples are given below.

Example 3

(a) *Aktie* 'stock' : *Kauf der Aktien* 'buying of stocks', *Aktien kaufen* 'to buy stocks', *Aktien an der Börse* 'stocks on the bourse' (is inappropriate because the word "bourse" describes a place and not an event)

(b) *beredt* 'eloquent': *beredt sprechen* 'to speak eloquently', *ein beredter Sprecher* 'an eloquent speaker' (is inappropriate because the word "speaker" describes a person and not an event)

The test was conducted via e-mail. In order to compare the human associations with the extracted AP-relations, we have manually annotated the obtained phrases with SALSA frames. The agreement for the described task for every cue word was calculated as the averaged pairwise agreement between the AP-relations delivered by the three subjects, S_1, S_2 and S_3, as follows:

$$Agr = \frac{\frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} + \frac{|S_2 \cap S_3|}{|S_2 \cup S_3|} + \frac{|S_2 \cap S_3|}{|S_2 \cup S_3|}}{3}.$$

Agreement results for every cue word are reported in table 2. Second column of the table contains gold standard predicates which were provided by all 3 participants treating the same word.[4] Averaging over all words, we got a mean agreement of 13%. Though this value seems to be low, it is consistent with a mean agreement of 11,8% for a similar task reported in (Cimiano and Wenderoth, 2007), see section 3. Cimiano and Wenderoth (2007) show that the lowest agreement is yielded for more abstract words, while the agreement for very concrete words is reasonable. We could not make a similar observation, see table 2.

Comparison with the Gold Standard

In the first experiment we checked whether predicates which people associate with the test arguments can be automatically extracted by our procedure. For this aim we compared the gold standard with all automatically extracted argument-predicate relations[5] containing some of the 30 cue words as follows. These relations were ranked according to the relatedness measure described in previous section. In line with (Cimiano and Wenderoth, 2007) we exploited an approach common in information retrieval for estimating the quality of correspondence of a ranked output to a gold standard, see (Baeza-Yates and Ribeiro-Neto, 1999).

Given some n automatically extracted relations with the highest ranking we calculated a precision-recall curve expressing precision and recall of our procedure compared to the gold standard. The precision characterizes the procedure exactness, i.e. how many redundant relations are retrieved. The

[4]The overall gold standard consists of 33 tuples.

[5]In order to evaluate the procedure extracting AP-relations on the basis of the semantic annotation we compared automatically extracted tuples to the gold standard tuples. For the procedure using the syntactic annotation only the AP-pairs were considered without regarding frames and FEs.

recall measures the completeness, i.e. how many relations of the gold standard are extracted automatically. For each point of the curve (which is a pair (p, r) of values of precision p and recall r) we calculated the F-measure as $F = 2pr/(p + r)$ which is the harmonic mean between recall and precision. The precision-recall curve is a set of precision values for the prespecified recall levels varying from 0 to 1 with a step 0,1. Then, to produce only one value evaluating the quality of the ranked output compared to the gold standard, for each precision-recall curve we calculated F_{max}, the maximal value of the F-measure achieved for the points of this curve. F_{max} expresses the best trade-off between precision and recall for the given ranked output. Finally, among all possible n (numbers of the considered relations with the highest ranking) we selected that one which provides the maximal F_{max} value.

The resulting maximal F_{max} values are 0,47 for the procedure extracting AP-relations on the basis of the semantic annotation and 0,41 for the procedure using the syntactic annotation only. We compared these results with the baseline results of maximal F_{max} values produced for the output with random ranking. The calculation of the baseline was repeated 100 times, each time a new random ranking was generated. The lowest baseline results are 0,08/0,06 (semantic/syntactic annotation), the highest are 0,18/0,14 and the medians are 0,1/0,07. One can see that the results produced using the relatedness measure (0,47/0,41) greatly exceed the baseline. Based on this comparison we conclude that the ranking done using the relatedness measure brings a significant advantage. The values of precision and recall for the reported maximal F_{max} values are 0,5/0,33 (semantic/syntactic annotation) and 0,45/0,54 respectively. This results show that half of the AP-relations from the gold standard appeared to be in the list of the top-ranked tuples extracted by the "semantic" procedure, while the size of this list ($n = 28$) was almost equal to the size of the gold standard (33). The differences in performance between the "semantic" and "syntactic" procedures could be explained by the fact that the "syntactic" procedure finds in the corpus more related predicates for every argument than the "semantic" one. Nevertheless, the "semantic" procedure shows better performance.

Next we investigated the results for each argument used in the gold standard separately in the same way as described above. For each argument the F_{max} measure has been computed. Because of the low agreement between the subjects questioned for the gold standard (see above), in these calculations we considered all predicates reported by our subjects. The calculated F_{max} values are reported in table 2 which shows a correlation between F_{max} values calculated for the "semantic" and "syntactic" procedures. However, there is no correlation with human agreement. This issue needs a further investigation, see section 7.

Human Judgments of the Relatedness

Following (Cimiano and Wenderoth, 2007), in order to check whether the calculated relatedness is reasonable according to human intuition, we have performed another experiment. For each of the 30 words selected for the gold standard we selected the 5 top ranked predicates. Since for some of the cue arguments only 3 predicates were found in the corpus, the final test set contains only 138 argument-predicate tuples. From these tuples we generated short grammatically correct phrases structurally similar to those in example 3. These phrases were uniformly distributed among 10 subjects so that every phrase was evaluated by one subject. The participants were asked to rate the phrases with respect to their naturalness using a scale from 0 to 3, whereby 0 means 'unnatural', 1 'possible', 2 'natural' and 3 'totally natural and self-evident'.

Further on we investigated the relationship between the human estimates and the relatedness values obtained automatically. For this aim we used the Spearman rank correlation coefficient. Because of four-points scale used, the human rankings are equal for many tuples which lead to the so-called effect of ties. For this reason we computed the correlation coefficient with a correction for ties. The coefficient value is 0,30 and this correlation is statistically significant with p-value 0,0006. Based on these results we conclude that our relatedness measure is correlated with human judgments. Taking into account the subjective character of human ranking in terms of naturalness, the achieved correlation values can be considered as high.

Table 2: Evaluation results for 30 gold standard cue words.

Cue word	**Shared predicates**	***Agr***	**Sem.** F_{max}	**Syn.** F_{max}
Name 'name'	*haben* 'to have'	14%	0,2	0,48
Urlaub 'vacation'	*fahren* 'to go'	8%	0,13	0,16
Sprache 'language'	*sprechen* 'to speak', *lernen* 'to learn'	14%	0,4	0,3
Strafe 'fine'	*verurteilen* 'to sentence'	11%	0,21	0,3
Stuhl 'chair'	*sitzen* 'to sit'	14%	0,1	0,2
Bombe 'bomb'	*hochgehen* 'to blow up'	14%	0,11	0,22
Blatt 'gazette', 'page', 'leaf'	–	2%	0	0
Flughafen 'airport'	*ankommen* 'to arrive', *fahren* 'to go'	21%	0,17	0,17
Gesetz 'low'	–	8%	0,17	0,38
Polizei 'police'	*rufen* 'to call'	11%	0,22	0,23
Kompromiss 'compromise'	*schliessen* 'to make'	15%	0,07	0,29
Fluggesellschaft 'airline'	–	3%	0,11	0,38
Antrag 'proposal', 'application'	*stellen* 'to introduce', *ablehnen* 'to decline'	24%	0,43	0,42
Zeitung 'newspaper'	*lesen* 'to read'	13%	0,17	0,09
Brief 'letter'	*verschicken* 'to send', *schreiben* 'to write'	19%	0,23	0,12
Flüchtling 'refugee'	*aufnehmen* 'to accept'	13%	0	0,07
Buch 'book'	*schreiben* 'to write', *lesen* 'to read'	15%	0,44	0,39
Zähler 'counter'	*ablesen* ' to read'	11%	0	0
Anzahl 'number'	–	3%	0,23	0,19
Prozent 'percent'	–	3%	0,48	0,21
Ziel 'goal'	*verfehlen* 'to miss', *erreichen* 'to reach'	20%	0,3	0,48
Schule 'school'	*schwänzen* 'to miss', *gehen* 'to go'	22%	0,13	0,23
Amt 'position', 'department'	*bekleiden, innehaben* 'to hold', *gehen* 'to go'	20%	0	0,17
Frage 'question'	*beantworten* 'to answer', *stellen* 'to ask'	20%	0,15	0,37
Mensch 'human'	*sein* 'to be'	16%	0,09	0,03
Zeuge 'witness'	*aussagen* 'to testify', *sein* 'to be'	22%	0,13	0,19
Thema 'theme'	–	7%	0,14	0,26
Preisträger 'prize winner'	–	5%	0,08	0,08
Initiative 'initiative'	*ergreifen* 'to take'	17%	0,1	0,13
Wohnung 'flat'	–	7%	0,09	0,17

7 Conclusion and Discussion

In this paper we presented an approach to automatic extraction of argument-predicate relations from a frame-annotated corpus.[6] In our approach we aimed to combine the advantages offered by annotated and unannotated lexical resources. Besides extracting AP-pairs the proposed method allows us to define types of semantic relations in terms of FrameNet frames. The proposed procedure is not restricted to arguments expressed by nouns and treats also other content parts of speech.

The main goal of this paper was to show that though manually annotated corpora usually have a relatively small size, they can be successfully exploited for the relation extraction. An obvious limitation of the presented approach is that it is bounded to manual annotations which are hard to obtain. However, since semantic annotations are useful for many different goals in linguistics and NLP, the number of reliable annotated corpora constantly grows.[7] Moreover, recently several tools have been developed which perform role annotation automatically, for example see (Erk and Pado, 2006). Therefore we believe that approaches using semantic annotation are valid and promising. In the future we plan to experiment with large role-annotated corpora for English such as PropBank (approx. 300 000 words, (Palmer et al., 2005)) and the FrameNet-annotated corpus provided by the FN project (more than 135 000 annotated sentences, (Ruppenhofer et al., 2006)). Since these corpora do not contain syntactic annotation, for extracting argument-predicate relations we will need to parse annotated sentences.

There are several ways to improve the proposed procedure. First, an implementation of a more advanced anaphora resolution algorithm treating pronominal as well as nominal anaphora should significantly raise the precision/recall characteristics. Second, splitting German compounds occurring in the corpus should provide additional evidence. We have treated such words as *Kunde* 'client' and *Privatkunde* 'private client' as different lexemes, while they are strongly related se-

[6] The complete list of the extracted AP-relations as well as the results of the experiment will be available online at *http://www.ikw.uni-osnabrueck.de/∼eovchinn/APrels/*.

[7] At present FrameNet annotated corpora are available for English, German and Spanish, see *http://framenet.icsi.berkeley.edu*.

mantically and information about predicates co-occurring with the second word could probably be used for describing the semantics of the first one. Concerning relatedness measure, additional corpus-based measures such as Web-based measures (Cimiano and Wenderoth, 2007) or measures based on syntactic relations (Pustejovsky et al., 1993) could appear to be useful for improving the ranking of the extracted relations.

The presented procedure was evaluated quantitatively against human judgments obtained experimentally. The participants of the experiment were asked to provide short phrases containing given cue words and predicates associated with these words as well as to rate phrases generated from the automatically extracted AP-relations. Concerning the first experiment, the low human agreement has shown that the proposed association task appeared to be difficult for the subjects. Nevertheless, the described learning procedure proved to extract intuitively reasonable relations.

The evaluation strategy presented in this paper on relies on the underlying assumptions (**A1** and **A2** in section 5) and is compatible with the other approaches to relation extraction, cf. (Cimiano and Wenderoth, 2007). However, it is plausible that human responses in the context of providing associated predicates for target words will differ from the responses in the experimental settings where subjects are asked to infer implicit predicates, e.g. to extend phrases containing implicit predicates. In the future we plan to implement a procedure making use of the extracted AP-relations which would automatically extend phrases containing implicit predicates. Then we intend to compare output results of the procedure with the human responses. Additionally, a study of a possible correspondence between human agreement on associated predicates and a semantic type of an argument (e.g. concrete/abstract, natural kind/artifact) should be performed on more test arguments.

Potential Applications

As already mentioned in the literature, see for example (Lapata and Lascarides, 2003), knowledge about implicit predicates could be potentially useful for a variety of NLP tasks such as language generation, information extraction, question answering or machine translation. Many applications of semantic relations in NLP are connected to paraphrasing or query expansion, see for example (Voorhees, 1994). Suppose that a search engine or a question answering system receives the query *schnelle Bombe* 'quick bomb'. Probably, in this case the user is interested in finding information about bombs that explode quickly rather then about bombs in general. Knowledge about predicates associated with the noun *Bombe* 'bomb' could be used for predicting a set of probable implicit predicates. For generation of the semantically and syntactically correct paraphrases it is sometimes not enough to guess the most probable argument-predicate pairs. Information about types of an argument-predicate relation could be helpful, i.e. which semantic and syntactic position does the argument fill in the argument structure of the predicate. For example, compare *eine Bombe explodiert schnell* 'a bomb explodes quickly' for *schnelle Bombe* with *ein Buch schnell lesen/schreiben* 'to read/write a book quickly' for *schnelles Buch* 'quick book'. In the first case the argument *Bombe* fills the subject position, while in the second case *Buch* fills the object position. Since FrameNet contains information about syntactic realization patterns for frame elements, representation of argument-predicate relations in terms of frames directly supports generation of semantically and syntactically correct paraphrases.

The described procedure could also support manual development of a lexical resource, providing evidence from corpora as well as the distributional information.

References

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison Wesley, Harlow, 1. aufl. edition.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*.

Ted Briscoe, Ann Copestake, and Bran Boguraev. 1990. Enjoy the paper: Lexical semantics via lexicology. In *Proceedings of the 13th International Conference on Computational Linguistics*, pages 42–47.

John Bullinaria and Joseph Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39(3):510–526.

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Pado, and Manfred Pinkal.

2006. The SALSA corpus: A German corpus resource for lexical semantics. In *Proceedings of LREC 2006*, pages 969–974.

Philipp Cimiano and Johanna Wenderoth. 2007. Automatic Acquisition of Ranked Qualia Structures from the Web. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 888–895.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by Latent Semantic Analysis. *American Society of Information Science*, 41(6):391–407.

Katrin Erk and Sebastian Pado. 2006. Shalmaneser - a flexible toolbox for semantic role assignment. In *Proceedings of LREC 2006*, Genoa, Italy.

Charles J. Fillmore. 1976. Frame semantics and the nature of language. In *Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech*, volume 280, pages 20–32.

Mirella Lapata and Alex Lascarides. 2003. A Probabilistic Account of Logical Metonymy. *Computational Linguistics*, 29(2):261–316.

Vivi Nastase, Jelber Sayyad-Shirabad, Marina Sokolova, and Stan Szpakowicz. 2006. Learning noun-modifier semantic relations with corpus-based and wordnet-based features. In *Proceedings of the AAAI 2006*.

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

James Pustejovsky, Peter Anick, and Sabine Bergler. 1993. Lexical semantic techniques for corpus analysis. *Computational Linguistics*, 19(2):331–358.

James Pustejovsky, Catherine Havasi, Roser Saur, Patrick Hanks, Anna Rumshisky, Jessica Littman, Jos Castao, and Marc Verhagen. 2006. Towards a generative lexical resource: The Brandeis Semantic Ontology. In *Proceedings of the Fifth Language Resource and Evaluation Conference*.

James Pustejovsky. 1991. The Generative Lexicon. *Computational Linguistics*, 17(4):409–441.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2006. FrameNet II: Extended Theory and Practice. *International Computer Science Institute*.

Ellen M. Voorhees. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 61–69.

Detecting Speculations and their Scopes in Scientific Text

Arzucan Özgür
Department of EECS
University of Michigan
Ann Arbor, MI 48109, USA
ozgur@umich.edu

Dragomir R. Radev
Department of EECS and
School of Information
University of Michigan
Ann Arbor, MI 48109, USA
radev@umich.edu

Abstract

Distinguishing speculative statements from factual ones is important for most biomedical text mining applications. We introduce an approach which is based on solving two sub-problems to identify speculative sentence fragments. The first sub-problem is identifying the speculation keywords in the sentences and the second one is resolving their linguistic scopes. We formulate the first sub-problem as a supervised classification task, where we classify the potential keywords as real speculation keywords or not by using a diverse set of linguistic features that represent the contexts of the keywords. After detecting the actual speculation keywords, we use the syntactic structures of the sentences to determine their scopes.

1 Introduction

Speculation, also known as hedging, is a frequently used language phenomenon in scientific articles, especially in experimental studies, which are common in the biomedical domain. When researchers are not completely certain about the inferred conclusions, they use speculative language to convey this uncertainty. Consider the following example sentences from abstracts of articles in the biomedical domain. The abstracts are available at the U.S. National Library of Medicine PubMed web page[1]. The PubMed Identifier (PMID) of the corresponding article is given in parenthesis.

1. *We showed that the Roaz protein bound specifically to O/E-1 by using the yeast two-hybrid system. (PMID: 9151733)*

2. *These data suggest that p56lck is physically associated with Fc gamma RIIIA (CD16) and functions to mediate signaling events related to the control of NK cellular cytotoxicity. (PMID: 8405050)*

The first sentence is definite, whereas the second one contains speculative information, which is conveyed by the use of the word *"suggest"*. While speculative information might still be useful for biomedical scientists, it is important that it is distinguished from the factual information.

Recognizing speculations in scientific text has gained interest in the recent years. Previous studies focus on identifying speculative sentences (Light et al., 2004; Medlock and Briscoe, 2007; Szarvas, 2008; Kilicoglu and Bergler, 2008). However, in many cases, not the entire sentence, but fragments of a sentence are speculative. Consider the following example sentences.

1. *The mature mitochondrial forms of the erythroid and housekeeping ALAS isozymes are predicted to have molecular weights of 59.5 kd and 64.6 kd, respectively. (PMID: 2050125)*

2. *Like RAD9, RAD9B associates with HUS1, RAD1, and RAD17, suggesting that it is a RAD9 paralog that engages in similar biochemical reactions. (PMID: 14611806)*

Both sentences are speculative, since they contain speculative information, which is signaled by the use of the word *"predicted"* in the first sentence and the word *"suggesting"* in the second sentence. The scope of the speculation keyword *"predicted"* in the first sentence spans the entire sentence. Therefore, classifying the sentence as speculative does not cause information loss. However, the scope of the speculation keyword *"suggesting"* in the second sentence applies only to the second clause of the sentence. In other words, only the statement *"RAD9B is a RAD9 paralog that engages in similar biochemical reactions"* is speculative. The statement *"Like RAD9, RAD9B associates with HUS1, RAD1, and RAD17"* conveys factual information. Therefore, classifying

[1] http://www.ncbi.nlm.nih.gov/pubmed/

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1398–1407,
Singapore, 6-7 August 2009.

the entire sentence as speculative will result in information loss.

In this paper, we aim to go beyond recognizing speculative sentences and tackle the problem of identifying speculative fragments of sentences. We propose an approach which is based on solving two sub-problems: (1) detecting the real speculation keywords, (2) resolving their linguistic scopes in the sentences. As the previous examples demonstrated speculations are signaled with speculation keywords (e.g. *might, suggest, likely, hypothesize, could, predict, and etc.*). However, these keywords are not always used in a speculative context. In other words, they are not always real speculation keywords. Unlike previous approaches which classify sentences as speculative or not, we formulate the problem as classifying the keywords as real speculation keywords or not. We extract a diverse set of features such as linguistic features that represent the context of the keyword and positional features of the sentence in which the keyword occurs. We use these features with Support Vector Machines (SVM) to learn models to classify whether the occurrence of a keyword is in a speculative context or not. After detecting the real speculation keywords, we use the syntactic structures of the sentences to identify their linguistic scopes.

2 Related Work

Although hedging in scientific articles has been studied from a linguistics perspective since the 1990s (e.g. (Hyland, 1998)), it has only gained interest from a natural language processing perspective in the recent years.

The problem of identifying speculative sentences in biomedical articles has been introduced by Light *et al.* (2004). The authors discussed the possible application areas of recognizing speculative language and investigated whether the notion of speculative sentences can be characterized to enable manual annotation. The authors developed two automated systems to classify sentences as speculative or not. The first method is based on substring matching. A sentence is classified as speculative if it contains one of the 14 predefined strings (*suggest, potential, likely, may, at least, in part, possibl, further investigation, unlikely, putative, insights, point toward, promise, propose*). The second method is based on using SVM with bag-of-words features. The substring matching method performed slightly better than the SVM with bag-of-words features approach.

Medlock and Briscoe (2007) extended the work of Light *et al.* (2004) by refining their annotation guidelines and creating a publicly available data set (FlyBase data set) for speculative sentence classification. They proposed a weakly supervised machine learning approach to classify sentences as speculative or not with the aim of minimizing the need for manually labeled training data. Their approach achieved 76% precision/recall break-even point (BEP) performance on the FlyBase data set, compared to the BEP of 60% obtained by Light *et al.*'s (2004) substring matching approach on the same data set. Szarvas (2008) extended the weakly supervised machine learning methodology of Medlock and Briscoe (2007) by applying feature selection to reduce the number of candidate keywords, by using limited manual supervision to filter the features, and by extending the feature representation with bigrams and trigrams. In addition, by following the annotation guidelines of Medlock and Briscoe (2007), Szarvas (2008) made available the BMC Bioinformatics data set, by annotating four full text papers from the open access BMC Bioinformatics website. They achieved a BEP performance of 85.29% and an F-measure of 85.08% on the FlyBase data set. The F-measure performance achieved on the BMC Bioinformatics data set was 74.93% when the FlyBase data set was used for training. Kilicoglu and Bergler (2008) compiled a list of speculation keywords from the examples in (Hyland, 1998) and extended this list by using WordNet (Fellbaum, 1998) and UMLS SPECIALIST Lexicon (McCray et al., 1994). They used manually crafted syntactic patterns to identify speculative sentences and achieved a BEP and an F-measure of 85% on the FlyBase data set and a BEP and an F-measure of 82% on the BMC Bioinformatics data set.

Unlike pervious studies, which treat the problem of identifying speculative language as a sentence classification task, we tackle the more challenging problem of identifying the portions of sentences which are speculative. In other words, we allow a sentence to include both speculative and non-speculative parts. We introduce and evaluate a diverse set of features that represent the context of a keyword and use these features in a supervised machine learning setting to classify

the keywords as real speculation keywords or not. Then, we develop a rule-based method to determine their linguistic scopes by considering the keyword-specific features and the syntactic structures of the sentences. To the best of our knowledge, the BioScope corpus (Vincze et al., 2008) is the only available data set that has been annotated for speculative sentence fragments and we report the first results on this corpus.

3 Corpus

The BioScope corpus[2] has been annotated at the token level for speculation keywords and at the sentence level for their linguistic scopes (Vincze et al., 2008). The corpus consists of three subcorpora: medical free texts (radiology reports), biomedical article abstracts, and biomedical full text articles. In this paper we focus on identifying speculations in scientific text. Therefore, we use the biomedical article abstracts and the biomedical full text articles in our experiments. The statistics (number of documents, number of sentences, and number of occurrences of speculation keywords) for these two sub-corpora are given in Table 1. The scientific abstracts in the BioScope corpus were included from the Genia corpus (Collier et al., 1999). The full text papers consist of five articles from the FlyBase data set and four articles from the open access BMC Bioinformatics website. The sentences in the FlyBase and BMC Bioinformatics data sets were annotated as speculative or not and made available by Medlock and Briscoe (2007) and Szarvas (2008), respectively and have been used by previous studies in identifying speculative sentences (Medlock and Briscoe, 2007; Kilicoglu and Bergler, 2008; Szarvas, 2008). Vincze *et al.* (2008) annotated these full text papers and the Genia abstracts for speculation keywords and their scopes and included them to the BioScope corpus. The keywords were annotated with a minimalist strategy. In other words, the minimal unit that expresses speculation was annotated as a keyword. A keyword can be a single word (e.g. suggest, predict, might) or a phrase (complex keyword), if none of the words constituting the phrase expresses a speculation by itself. For example the phrase *"no evidence of"* in the sentence *"Direct sequencing of the viral genomes and reinfection kinetics showed no evidence of wild-type reversion even after prolonged infection with the Tat- virus."* is an example of a complex keyword, since the words forming the phrase can only express speculation together.

Data Set	Documents	Sentences	Hedge Keywords
Abstracts	1273	11871	2694
Full Papers	9	2670	682

Table 1: Summary of the biomedical scientific articles subcorpora of the BioScope corpus

In contrast to the minimalist strategy followed when annotating the keywords, the annotation of scopes of the keywords was performed by assigning the scope to the largest syntactic unit possible by including all the elements between the keyword and the target word to the scope (in order to avoid scopes without a keyword) and by including the modifiers of the target word to the scope (Vincze et al., 2008). The reader can refer to (Vincze et al., 2008) for the details of the corpus and the annotation guidelines.

The inter-annotator agreement rate was measured as the F-measure of the annotations of the first annotator by considering the annotations of the second one as the gold standard. The agreement rate for speculation keyword annotation is reported as 92.05% for the abstracts and 90.81% for the full text articles and the agreement rate for speculation scope resolution is reported as 94.04% for the abstracts and 89.67% for the full text articles (Vincze et al., 2008). These rates can be considered as the upper bounds for the automated methods proposed in this paper.

4 Identifying Speculation Keywords

Words and phrases such as *"might", "suggest", "likely", "no evidence of", and "remains to be elucidated"* that can render statements speculative are called speculation keywords. Speculation keywords are not always used in speculative context. For instance, consider the following sentences:

1. *Thus, it appears that the T-cell-specific activation of the proenkephalin promoter is mediated by NF-kappa B. (PMID: 91117203)*

2. *Differentiation assays using water soluble phorbol esters reveal that differentiation becomes irreversible soon after AP-1 appears. (PMID: 92088960)*

The keyword *"appears"* in the first sentence renders it speculative. However, in the second sentence, *"appears"* is not used in a speculative context.

[2] Available at: http://www.inf.u-szeged.hu/rgai/bioscope

The first sub-problem that we need to solve in order to identify speculative sentence fragments is identifying the real speculation keywords in a sentence (i.e. the keywords which convey speculative meaning in the sentence). We formulate the problem as a supervised classification task. We extract the list of keywords from the training data which has been labeled for speculation keywords. We match this list of keywords in the unlabeled (test data) and train a model to classify each occurrence of a keyword in the unlabeled test set as a real speculation keyword or not. The challenge of the task can be demonstrated by the following statistics from the Genia Abstracts of the BioScope corpus. There are 1273 abstracts in the corpus. There are 138 unique speculation keywords and the total number of their occurrence in the abstracts is 6125. In only 2694 (less than 50%) of their occurrences they are used in speculative context (i.e., are real speculation keywords).

In this study we focus on identifying the features that represent the context of a speculation keyword and use SVM with linear kernel (we used the SVM^{light} package (Joachims, 1999)) as our classification algorithm. The following subsection describes the set of features that we propose.

4.1 Feature Extraction

We introduce a set of diverse types of features including keyword specific features such as the stem and the part-of-speech (POS) of the keyword, and keyword context features such as the words surrounding the keyword, the dependency relation types originating at the keyword, the other keywords that occur in the same sentence as the keyword, and positional features such as the section of the paper in which the keyword occurs. While designing the features, we were inspired by studies on other natural language processing problems such as Word Sense Disambiguation (WSD) and summarization. For example, machine learning methods with features based on part-of-speech tags, word stems, surrounding and co-occurring words, and dependency relationships have been successfully used in WSD (Montoyo et al., 2005; Ng and Lee, 1996; Dligach and Palmer, 2008) and positional features such as the position of a sentence in the document have been used in text summarization (e.g. (Radev et al., 2004)).

4.1.1 Keyword Features

Statistics from the BioScope corpus suggest that different keywords have different likelihoods of being used in a speculative context (Vincze et al., 2008). For example, the keyword *"suggest"* has been used in a speculative context in all its occurrences in the abstracts and in the full papers. On the other hand, *"appear"* is a real speculation keyword in 86% of its occurrences in the abstracts and in 83% of its occurrences in the full papers, whereas *"can"* is a real speculation keyword in 12% of its occurrences in the abstracts and in 16% of its occurrences in the full papers. POS of a keyword might also play a role in determining whether it is a real speculation keyword or not. For example, consider the keyword *"can"*. It is more likely to have been used in a speculative context when it is a modal verb, than when it is a noun. Based on these observations, we hypothesize that features specific to a keyword such as the keyword itself, the stem of the keyword, and the POS of the keyword might be useful in discriminating the speculative versus non-speculative use of it. We use Porter's Stemming Algorithm (Porter, 1980) to obtain the stems of the keywords and Stanford Parser (de Marneffe et al., 2006) to get the POS of the keywords. If a keywords consists of multiple words, we use the concatenation of the POS of the words constituting the keyword as a feature. For example, the extracted POS feature for the keywords *"no evidence"* and *"no proof"* is *"DT.NN"* .

4.1.2 Dependency Relation Features

Besides the occurrence of a speculation keyword, the syntactic structure of the sentence also plays an important role in characterizing speculations. Kilicoglu and Bergler (2008) showed that manually identified syntactic patterns are effective in classifying sentences as speculative or not. They identified that, while some keywords do not indicate hedging when used alone, they might act as good indicators of hedging when used with a clausal complement or with an infinitival clause. For example, the *"appears"* keyword in the example sentences, which are given in the beginning of Section 4, is not a real speculation keyword in the second example *"...soon after AP-1 appears."* , whereas it is a real speculation keyword in the first example, where it is used with a *that* clausal complement *"...it appears that..."*. Similarly, *"appears"* is used in a speculative context in the fol-

lowing sentence, where it is used with an infinitival clause: *"Synergistic transactivation of the BMRF1 promoter by the Z/c-myb combination appears to involve direct binding by the Z protein."*.

Another observation is that, some keywords act as real speculation keywords only when used with a negation. For example, words such as *"know", "evidence", and "proof"* express certainty when used alone, but express a speculation when used with a negation (e.g., *"not known", "no evidence", "no proof"*).

Auxiliaries in verbal elements might also give clues for the speculative meaning of the main verbs. Consider the example sentence: *"Our findings may indicate the presence of a reactivated virus hosted in these cells."*. The modal auxiliary *"may"* acts as a clue for the speculative context of the main verb *"indicate"*.

We defined boolean features to represent the syntactic structures of the contexts of the keywords. We used the Stanford Dependency Parser (de Marneffe et al., 2006) to parse the sentences that contain a candidate speculation keyword and extracted the following features from the dependency parse trees.

Clausal Complement: A Boolean feature which is set to 1, if the keyword has a child which is connected to it with a clausal complement or infinitival clause dependency type.

Negation: A Boolean feature which is set to 1, if the keyword (1) has a child which is connected to it with a negation dependency type (e.g. "not known": "not" is a child of "known", and the Stanford Dependency Type connecting them is "neg") or (2) the determiner "no" is a child of the keyword (e.g., "no evidence": "no" is a child of "evidence" and the Stanford Dependency Type connecting them is "det").

Auxiliary: A Boolean feature which is set to 1, if the keyword has a child which is connected to it with an auxiliary dependency type (e.g. "may indicate": "may" is a child of "indicate", and the Stanford Dependency Type connecting them is "aux").

If a keyword consists of multiple-words, we examine the children of the word which is the ancestor of the other words constituting the keyword. For example, "no evidence" is a multi-word keyword, where "evidence" is the parent of "no". Therefore, we extract the dependency parse tree features for the word "evidence".

4.1.3 Surrounding Words

Recent studies showed that using machine learning with variants of the "bag-of-words" feature representation is effective in classifying sentences as speculative vs. non-speculative (Light et al., 2004; Medlock and Briscoe, 2007; Szarvas, 2008). Therefore, we also decided to include bag-of-words features that represent the context of the speculation keyword. We extracted the words surrounding the keyword and performed experiments both with and without stemming, and with window sizes of one, two, and three. Consider the sentence: *"Our findings may indicate the presence of a reactivated virus hosted in these cells."*. The bag-of-words features for the keyword "indicate", when a window size of three and no stemming is used are: *"our", "findings", "may", "indicate", "the", "presence", "of"*. In other words, the feature set consists of the keyword, the three words to the left of the keyword, and the three words to the right of the keyword.

4.1.4 Positional Features

Different parts of a scientific article might have different characteristics in terms of the usage of speculative language. For example, Hyland (1998) analyzed a data set of molecular biology articles and reported that the distribution of speculations is similar between abstracts and full text articles, whereas the Results and Discussion sections tend to contain more speculative statements compared to the other sections (e.g. Materials and Methods or Introduction and Background sections). The analysis of Light *et al.* (2004) showed that the last sentence of an abstract is more likely to be speculative than non-speculative.

For the scientific abstracts data set, we defined the following boolean features to represent the position of the sentence the keyword occurs in. Our intuition is that titles and the first sentences in the abstract tend to be non-speculative, whereas the last sentence of the abstract tends to be speculative.

Title: A Boolean feature which is set to 1, if the keyword occurs in the title.

First Sentence: A Boolean feature which is set to 1, if the keyword occurs in the first sentence of the abstract.

Last Sentence: A Boolean feature which is set to 1, if the keyword occurs in the last sentence of the abstract.

For the scientific full text articles data set, we defined the following features that represent the position of the sentence in which the keyword occurs. Our assumption is that the "Results and Discussion" and the "Conclusion" sections tend to

contain more speculative statements than the "Materials and Methods" and "Introduction and Background" sections. We also assume that figure and table legends are not likely to contain speculative statements.

Title: A Boolean feature which is set to 1, if the keyword occurs in the title of the article, or in the title of a section or sub-section.

First Sentence: A Boolean feature which is set to 1, if the keyword occurs in the first sentence of the abstract.

Last Sentence: A Boolean feature which is set to 1, if the keyword occurs in the last sentence of the abstract.

Background: A Boolean feature which is set to 1, if the keyword occurs in the Background or Introduction section.

Results: A Boolean feature which is set to 1, if the keyword occurs in the Results or in the Discussion section.

Methods: A Boolean feature which is set to 1, if the keyword occurs in the Materials and Methods section.

Conclusion: A Boolean feature which is set to 1, if the keyword occurs in the Conclusion section.

Legend: A Boolean feature which is set to 1, if the keyword occurs in a table or figure legend.

4.1.5 Co-occurring Keywords

Speculation keywords usually co-occur in the sentences. Consider the sentence: *"We, therefore, wished to determine whether T3SO4 could mimic the action of thyroid hormone in vitro."*. Here, *"whether"* and *"could"* are speculation keywords and their co-occurence might be a clue for their speculative context. Therefore, we decided to include the co-occurring keywords to the feature set of a keyword.

5 Resolving the Scope of a Speculation

After identifying the real speculation keywords, the next step is determining their scopes in the sentences, so that the speculative sentence fragments can be detected. Manual analysis of sample sentences from the BioScope corpus and their parse trees suggests that the scope of a keyword can be characterized by its part-of-speech and the syntactic structure of the sentence in which it occurs. Consider the example sentence whose parse tree is shown in Figure 1. The sentence contains three speculation keywords, "or" and two occurrences of "might". The scope of the conjunction "or", extends to the "VP" whose children it coordinates. In other words, the scope of "or" is "[might be one of the earliest crucial steps in the lysis of normal and dex-resistant CEM cells, *or* might serve as a marker for the process]". Here, "or" conveys a speculative meaning, since we are not certain which of the two sub-clauses (sub-clause 1: [might be one of the earliest crucial steps in the lysis of normal and dex-resistant CEM cells] *or* sub-clause 2: [might serve as a marker for the process]) is correct. The scope of both occurrences of the modal verb "might" is the parent "VP". In other words, the scope of the first occurrence of "might" is "[*might* be one of the earliest crucial steps in the lysis of normal and dex-resistant CEM cells]" and the scope of the second occurrence of "might" is "[*might* serve as a marker for the process]". By examining the keywords, sample sentences and their syntactic parse trees we developed the following rule-based approach to resolve the scopes of speculation keywords. The examples given in this section are based on the syntactic structure of the Penn Tree Bank. But, the rules are generic (e.g. "the scope of a verb followed by an infinitival clause, extends to the whole sentence").

The scope of a conjunction or a determiner (e.g. or, and/or, vs) is the syntactic phrase to which it is attached. For example, the scope of "or" in Figure 1 is the "VP" immediately dominating the "CC".

The scope of a modal verb (e.g. may, might, could) is the "VP" to which it is attached. For example, the scope of "might" in Figure 1 is the "VP" immediately dominating the "MD".

The scope of an adjective or an adverb starts with the keyword and ends with the last token of the highest level "NP" which dominates the adjective or the adverb. Consider the sentence "The endocrine events that are rapidly expressed (seconds) are due to a [possible interaction with cellular membrane]." The scope of the speculation keyword "possible" is enclosed in rectangular brackets. The sub-tree that this scope maps to is: "(NP (NP (DT a) (JJ possible) (NN interaction)) (PP (IN with) (NP (JJ cellular) (NN membrane))))". If there does not exist a "NP" dominating the adverb or adjective keyword, the scope extends to the whole sentence. For example the scope of the speculation adverb "probably" in the sentence "[The remaining portion of the ZFB motif was probably lost in TPases of insect Transib transposons]" is the whole sentence.

The scope of a verb followed by an infinitival

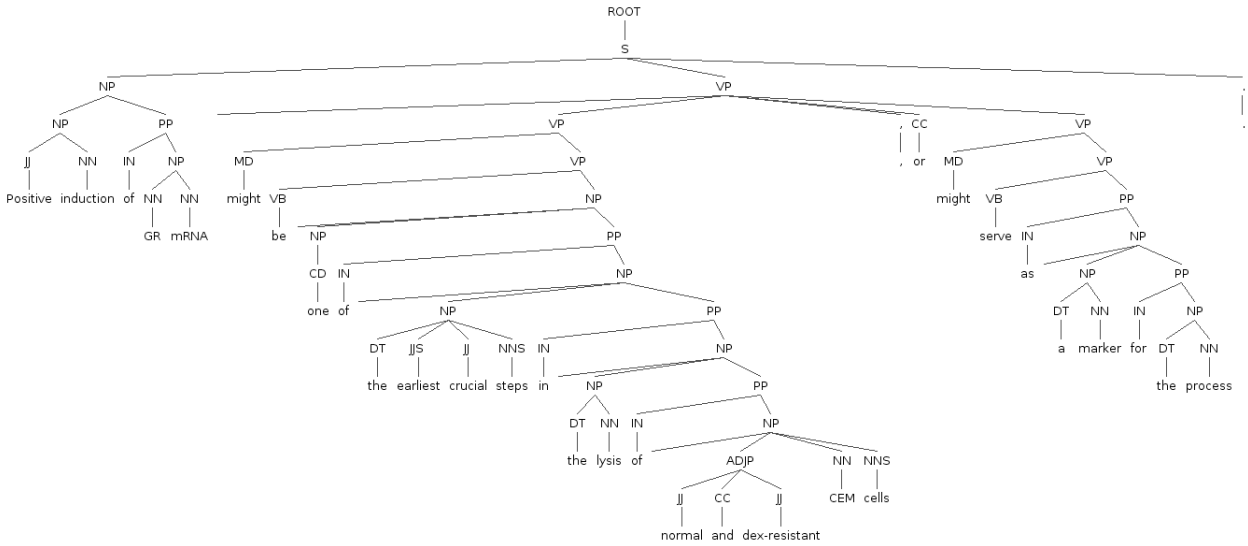


Figure 1: The syntactic parse tree of the sentence "*Positive induction of GR mRNA might be one of the earliest crucial steps in the lysis of normal and dex-resistant CEM cells, or might serve as a marker for the process.*"

clause extends to the whole sentence. For example, the scope of the verb "appears" followed by the "to" infinitival clause is the whole sentence in "[The block of pupariation appears to involve signaling through the adenosine receptor (AdoR)]".

The scope of a verb in passive voice extends to the whole sentence such as the scope of "suggested" in "[The existence of such an independent mechanism has also been suggested in mammals]".

If none of the above rules apply, the scope of a keyword starts with the keyword and ends at the end of the sentence (or clause). An example is the scope of "suggested" in "This [suggested that there is insufficient data currently available to determine a reliable ratio for human]".

6 Evaluation

We evaluated our approach on two different types of scientific text from the biomedical domain, namely the scientific abstracts sub-corpus and the full text articles sub-corpus of the BioScope corpus (see Section 3). We used stratified 10-fold cross-validation to evaluate the performance on the abstracts. In each fold, 90% of the abstracts are used for training and 10% are used to test. To facilitate comparison with future studies the PubMed Identifiers of the abstracts that we used as a test set in each fold are provided[3]. The full text papers sub-corpus consists of nine articles. We used leave-one-out cross-validation to evaluate the performance on the full text papers. In each iteration eight articles are used for training and one article is used to test. We report the average results over the runs for each data set.

6.1 Evaluation of Identifying Speculation Keywords

To classify whether the occurrence of a keyword is in speculative context or not, we built linear SVM models by using various combinations of the features introduced in Section 4.1. Tables 2 and 3 summarize the results obtained for the abstracts and the full text papers, respectively. *BOW N* is the bag-of-words features obtained from the words surrounding the keyword (see Section 4.1.3). N is the window size. We experimented both with the stemmed and non-stemmed versions of this feature type. The non-stemmed versions performed slightly better than the stemmed versions. The reason might be due to the different likelihoods of being used in a speculative context of different inflected forms of words. For example, consider the words "appears" and "appearance". They have the same stems, but "appearance" is less likely to be a real speculation keyword than "appears". Another observation is that, decreasing the window size led to improvement in performance. This suggests that the words right before and right after the candidate speculation keyword are more effective in distinguishing its speculative vs. non-speculative context compared to a wider local context. Wider local context might create sparse data and degrade

[3] http://belobog.si.umich.edu/clair/bioscope/

performance. Consider the example, "it appears that TP53 interacts with AR". The keyword "appears", and BOW1 ("it" and "that") are more relevant for the speculative context of the keyword than "TP53", "interacts", and "with". Therefore, for the rest of the experiments we used the *BOW 1* version, i.e., the non-stemmed surrounding bag-of-words with window size of 1. *KW* stands for the keyword specific features, i.e., the keyword, its stem, and its part-of-speech (discussed in Section 4.1.1). *DEP* stands for the dependency relation features (discussed in Section 4.1.2). *POS* stands for the positional features (discussed in Section 4.1.4) and *CO-KW* stands for the co-occurring keywords feature (discussed in Section 4.1.5).

Our results are not directly comparable with the prior studies about identifying speculative sentences (see Section 2), since we attempted to solve a different problem, which is identifying speculative parts of sentences. Only the substring matching approach that was introduced in (Light et al., 2004) could be adapted as a keyword classification task, since the substrings are keywords themselves and we used this approach as a baseline in the keyword classification sub-problem. We compare the performances of our models with two baseline methods, which are based on the substring matching approach. Light *et al.* (2004) have shown that the substring matching method with a predefined set of 14 strings performs slightly better than an SVM model with bag-of-words features in classifying sentences as speculative vs. non-speculative (see Section 2). In baseline 1, we use the 14 strings identified in (Light et al., 2004) and classify all the keywords in the test set that match any of them as real speculation keywords. Baseline 2 is similar to baseline 1, with the difference that rather than using the set of strings in (Light et al., 2004), we extract the set of keywords from the training set and classify all the words (or phrases) in the test set that match any of the keywords in the list as real speculation keywords.

Baseline 1 achieves high precision, but low recall. Whereas, baseline 2 achieves high recall in the expense of low precision. All the SVM models in Tables 2 and 3 achieve more balanced precision and recall values, with F-measure values significantly higher than the baseline methods. We start with a model that uses only the keyword-specific features (KW). This type of feature alone achieved a significantly better performance than the baseline methods (90.61% F-measure for the abstracts and 80.57% F-measure for the full text papers), suggesting that the keyword-specific features are important in determining its speculative context. We extended the feature set by including the dependency relation (DEP), surrounding words (BOW 1), positional (POS), and co-occurring keywords (CO-KW) features. Each new type of included feature improved the performance of the model for the abstracts. The best F-measure (91.69%) is achieved by using all the proposed types of features. This performance is close to the upper bound, which is the human inter-annotator agreement F-measure of 92.05%.

Including the co-occurring keywords to the feature set for full text articles slightly improved precision, but deceased recall, which led to lower F-measure. The best F-measure (82.82%) for the full text articles is achieved by using all the feature types except the co-occurring keywords. The achieved performance is significantly higher than the baseline methods, but lower than the human inter-annotator agreement F-measure of 90.81%. The lower performance for the full text papers might be due to the small size of the data set (9 full text papers compared to 1273 abstracts).

Method	Recall	Precision	F-Measure
Baseline 1	52.84	92.71	67.25
Baseline 2	97.54	43.66	60.30
BOW 3 - stemmed	81.47	92.36	86.51
BOW 2 - stemmed	81.56	93.29	86.97
BOW 1 - stemmed	83.08	93.83	88.05
BOW 3	82.58	92.04	86.98
BOW 2	82.77	92.74	87.41
BOW 1	83.27	93.67	88.10
KW: kw, kw-stem, kw-pos	88.62	92.77	90.61
KW, DEP	88.77	92.67	90.64
KW, DEP, BOW 1	88.46	94.71	91.43
KW, DEP, BOW 1, POS	88.16	95.21	91.50
KW, DEP, BOW 1, POS, CO-KW	88.22	95.56	91.69

Table 2: Results for the Scientific Abstracts

Method	Recall	Precision	F-Measure
Baseline 1	33.77	86.75	47.13
Baseline 2	88.22	52.57	64.70
BOW 3 - stemmed	70.79	83.88	76.58
BOW 2 - stemmed	72.31	85.49	78.11
BOW 1 - stemmed	73.49	84.35	78.41
BOW 3	70.54	82.56	75.88
BOW 2	71.52	85.93	77.94
BOW 1	73.72	86.27	79.43
KW: kw, kw-stem, kw-pos	75.21	87.08	80.57
KW, DEP	75.02	89.49	81.53
KW, DEP, BOW 1	76.15	89.54	82.27
KW, DEP, BOW 1, POS	76.17	90.81	82.82
KW, DEP, BOW 1, POS, CO-KW	75.76	90.82	82.58

Table 3: Results for the Scientific Full Text Papers

6.2 Evaluation of Resolving the Scope of a Speculation

We compared the proposed rule-based approach for scope resolution with two baseline methods. Previous studies classify sentences as speculative or not, therefore implicitly assigning the scope of a speculation to the whole sentence (Light et al., 2004; Medlock and Briscoe, 2007; Szarvas, 2008; Kilicoglu and Bergler, 2008). Baseline 1 follows this approach and assigns the scope of a speculation keyword to the whole sentence. Szarvas (2008) suggest assigning the scope of a keyword from its occurrence to the end of the sentence. They state that this approach works accurately for clinical free texts, but no any results are reported (Szarvas, 2008). Baseline 2 follows the approach proposed in (Szarvas, 2008) and assigns the scope of a keyword to the fragment of the sentence that starts with the keyword and ends at the end of the sentence. Table 4 summarizes the accuracy results obtained for the abstracts and the full text papers.

The poor performance of baseline 1, emphasizes the importance of detecting the portions of sentences that are speculative, since less than 5% of the sentences that contain speculation keywords are entirely speculative. Classifying the entire sentences as speculative or not leads to loss in information for more than 95% of the sentences. The rule-based method significantly outperformed the two baseline methods, indicating that the part-of-speech of the keywords and the syntactic parses of the sentences are effective in characterizing the speculation scopes.

Method	Accuracy-Abstracts	Accuracy-Full text
Baseline 1	4.82	4.29
Baseline 2	67.60	42.82
Rule-based method	79.89	61.13

Table 4: Scope resolution results

7 Conclusion

We presented an approach to identify speculative sentence fragments in scientific articles. Our approach is based on solving two sub-problems. The first one is identifying the keywords which are used in speculative context and the second one is determining the scopes of these keywords in the sentences. We evaluated our approach for two types of scientific texts, namely abstracts and full text papers from the BioScope corpus.

We formulated the first sub-problem as a supervised classification task, where the aim is to learn models to classify the candidate speculation keywords as real speculation keywords or not. We focused on identifying different types of linguistic features that capture the contexts of the keywords. We achieved a performance which is significantly better than the baseline methods and comparable to the upper-bound, which is the human inter-annotator agreement F-measure.

We hypothesized that the scope of a speculation keyword can be characterized by its part-of-speech and the syntactic structure of the sentence and developed rules to map the scope of a keyword to the nodes in the syntactic parse tree. We achieved a significantly better performance compared to the baseline methods. The considerably lower performance of the baseline of assigning the scope of a speculation keyword to the whole sentence indicates the importance of detecting speculative sentence portions rather than classifying the entire sentences as speculative or not.

Acknowledgements

This work was supported in part by the NIH Grant U54 DA021519 to the National Center for Integrative Biomedical Informatics.

References

Nigel Collier, Hyun S. Park, Norihiro Ogata, Yuka Tateishi, Chikashi Nobata, Tomoko Ohta, Tateshi Sekimizu, Hisao Imai, Katsutoshi Ibushi, and Jun I. Tsujii. 1999. The GENIA project: corpus-based knowledge acquisition and information extraction from genome research papers. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 271–272. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of LREC-06*.

Dmitriy Dligach and Martha Palmer. 2008. Novel Semantic Features for Verb Sense Disambiguation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press.

Ken Hyland. 1998. *Hedging in Scientific Research Articles*. John Benjamins Publishing Co.

T. Joachims, 1999. *Advances in Kernel Methods-Support Vector Learning*, chapter Making Large-Scale SVM Learning Practical. MIT-Press.

Halil Kilicoglu and Sabine Bergler. 2008. Recognizing speculative language in biomedical research articles: a linguistically motivated perspective. *BMC Bioinformatics*, 9(Suppl 11).

Marc Light, Xin Ying Qiu, and Padmini Srinivasan. 2004. The language of bioscience: Facts, speculations, and statements in between. In Lynette Hirschman and James Pustejovsky, editors, *HLT-NAACL 2004 Workshop: BioLINK 2004, Linking Biological Literature, Ontologies and Databases*, pages 17–24, Boston, Massachusetts, USA, May 6. Association for Computational Linguistics.

A. T. McCray, S. Srinivasan, and A. C. Browne. 1994. Lexical methods for managing variation in biomedical terminologies. *Proc Annu Symp Comput Appl Med Care*, pages 235–239.

Ben Medlock and Ted Briscoe. 2007. Weakly supervised learning for hedge classification in scientific literature. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 992–999, Prague, Czech Republic, June. Association for Computational Linguistics.

Andres Montoyo, Armando Suarez, German Rigau, and Manuel Palomar. 2005. Combining knowledge- and corpus-based word-sense-disambiguation methods. *Journal of Artificial Intelligence Research*, 23:299–330.

H. T. Ng and H. B Lee. 1996. Integrating multiple knowledge sources to disambiguate word senses: An exemplar-based approach. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 3(14):130–137.

Dragomir R. Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Adam Winkel, and Zhang Zhu. 2004. Mead - a platform for multidocument multilingual text summarization. In *Proceedings of LREC 2004*.

Gyorgy Szarvas. 2008. Hedge classification in biomedical texts with a weakly supervised selection of keywords. In *ACL 2008*.

Veronika Vincze, Gyorgy Szarvas, Richard Farkas, Gyorgy Mora, and Janos Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, 9(Suppl 11).

Cross-Cultural Analysis of Blogs and Forums with Mixed-Collection Topic Models

Michael Paul and Roxana Girju
University of Illinois at Urbana Champaign
Urbana, IL 61801
{mjpaul2, girju}@illinois.edu

Abstract

This paper presents preliminary results on the detection of cultural differences from people's experiences in various countries from two perspectives: tourists and locals. Our approach is to develop probabilistic models that would provide a good framework for such studies. Thus, we propose here a new model, **ccLDA**, which extends over the Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and cross-collection mixture (ccMix) (Zhai et al., 2004) models on blogs and forums. We also provide a qualitative and quantitative analysis of the model on the cross-cultural data.

1 Introduction

In today's society, people from different cultural backgrounds have to understand each other, interact on a daily base and travel to or work in more than one country. Understanding cultural diversity, as well as addressing the need to communicate effectively across cultural divides, have become imperative in almost every aspect of life. These constitute an important language aspect since the lack of such cultural awareness can lead to misinterpretations.

This paper presents preliminary results on the detection of cultural differences from people's experiences in various countries from two perspectives: tourists and locals. Since the advent of Web 2.0, user-generated data in the form of blogs and newsgroup messages have reached high proportions. In this paper we take advantage of such resources of blogs and forums to perform various cross-cultural analyses.

Our approach is to develop probabilistic models that would provide a good framework for such studies. Thus, we propose here a new model, **ccLDA**, which extends over the Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and cross-collection mixture (ccMix) (Zhai et al., 2004) models. Our contribution is as follows:

(1) Unsupervised topic models such as LDA are elegant and flexible approaches to clustering large collections of unannotated data. These models, however, have conceptually focused on one single collection of text which is inadequate for comparative analyses of text.

We thus develop an LDA-based model that can not only discover topics but also model their similarities and differences across multiple text collections.

(2) We improve on similar previous work by crafting a model that can better generalize data and is less reliant on user-defined parameters.

(3) We apply our new model on blogs and forums to identify cross-cultural differences.

Thus, different models can be compared to reflect different hypotheses about the data.

The paper is organized as follows. In Section 2 we summarize relevant previous work and give a detailed description of the model in Section 3. Section 4 details the model's parameter estimation. Experimental results are presented in Section 5, followed by discussion, future work, and conclusions.

2 Previous Work

A topic model for comparing text collections (ccMix) was previously introduced by Zhai et al. (2004) for a problem called comparative text mining (CTM). Given news articles from different sources (about the same event), ccMix can extract what is common to all the sources and what is unique to one specific source.

Our model improves over ccMix by replacing their probabilistic latent semantic indexing (pLSI) (Hofmann, 1999) framework with that of LDA.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1408–1417,
Singapore, 6-7 August 2009.

Under the ccMix model, the probability of generating the ith word in a document belonging to collection c is:

$$P(w_i) = (1-\lambda_B)\sum_{z\in Z} P(z)(\lambda_C P(w_i|z) + (1-\lambda_C)P(w_i|z,c)) + \lambda_B P(w_i|B),$$

where each topic is denoted z. λ_B is the probability of choosing a word from the background word distribution and is user-defined. λ_C is also defined by the user and is the probability of drawing a word from the collection-independent word distribution instead of the collection-specific distribution. The parameters can be estimated using the Expectation-Maximization algorithm (Dempster et al., 1977).

However, in addition to the advantages of LDA over pLSI such as the incorporation of Dirichlet priors and a natural way to deal with new documents, our model avoids the limitations of using a single user-defined parameter λ_C – this probability is learned automatically under our model. Furthermore, we allow this probability to depend on the collection and topic, which is a less restrictive assumption.

Our model, ccLDA, shares with the LDA-Collocation (Griffiths et al., 2007) and Topical N-Grams (Wang et al., 2007) models the assumption that each word can come from two different word distributions, one of which depends on another observable variable. In these models, a word can come from either its topic's word distribution, or it can come from a word distribution associated with the previous word, in the case that the word is determined to be part of a collocation. The key difference here is that in these models, the alternative word distribution depends on the word preceding a token, while in ccLDA, this depends on the document's collection.

The model is also related to hierarchical variants of LDA, in particular the hierarchical Pachinko allocation (hPAM) (Mimno et al., 2007) model, in which both a topic and hierarchy depth are chosen, and there is a different word distribution at different levels in the hierarchy. A natural way to view our model is as a two-level hierarchy where the top level represents the collection-independent distributions and the bottom level represents the collection-specific distributions. One of the main differences here is that the discovered hierarchies in hPAM can be arbitrary, whereas the graphical structure of our model is pre-determined such that each topic has exactly one "sub-topic" representing each collection.

Wang et al. recently introduced Markov topic models (MTM) (2009), a family of models which can simultaneously learn the topic structure of a single collection while discovering correlated topics in other collections. This is promising in that this type of model makes no assentation that each topic is in some way shared across all collections. However, it does not explicitly model the similarities and differences between collections as we do in this research.

In computational linguistics, topic models have been used in various applications, such as predicting response to political webposts (Yano et al., 2009), analyzing Enron and academic emails (McCallum et al., 2007a), analyzing voting records and corresponding text of resolutions from the U.S. Senate and the U.N. (McCallum et al., 2007b), as well as studying the history of ideas in various research fields (Hall et al., 2008; Paul and Girju, 2009). To our knowledge, the application of topic models to identifying cross-cultural differences is novel.

3 The Model

In this section we first review the basic pLSI and LDA models. We then introduce our extension to LDA: *cross-collection LDA* (ccLDA).

3.1 Basic Topic Modeling

The most basic generative model that assumes document topicality is the standard Naïve Bayes model, where each document is assumed to belong to exactly one topic, and each topic is associated with a probability distribution over words (Mitchell, 1997).

While this single-topic approach can be sufficient for classification tasks, it is often too limiting for unsupervised grouping of semantically related words into topics. A better assumption is that each document is a mixture of topics. For example, a news article about a natural disaster may include topics about the causes of such disasters, the damage/death toll, and relief aid/efforts. Probabilistic latent semantic indexing (pLSI) (Hofmann, 1999) is one such model. Under this model, the probability of seeing the ith word in a document is:

$$P(w_i|d) = \sum_{z\in Z} P(w_i|z)P(z|d)$$

One of the main criticisms of pLSI is that each document is represented as a variable d and it is not clear how to label previously unseen documents. This issue is addressed by Blei et al. with latent Dirichlet allocation (2003). Furthermore, the probabilities under this model have Dirichlet priors, which results in more reasonable mixtures and less overfitting. In LDA, a document is generated as follows:

1) Draw a multinomial distribution of words ϕ_z from Dirichlet(β) for each topic z

2) For each document d[1], draw a topic mixture distribution $\theta^{(d)}$ from Dirichlet(α). Then for each word w_i in d:

a) Sample a topic z_i from $\theta^{(d)}$

b) Sample a word w_i from ϕ_z

The Dirichlet parameters α and β are vectors which represent the average of the respective distributions. In many applications, it is sufficient to assume that these vectors are uniform and to fix them at a value pre-defined by the user. In this case, the Dirichlet priors simply function as smoothing factors.

3.2 Cross-Collection LDA

In this subsection we introduce our extension of LDA for comparing multiple text collections, which we refer to as cross-collection LDA (ccLDA). Under this model, each topic is associated with two classes of word distributions: one that is shared among all collections, and one that is unique to the collection from which the document comes. For example, when modeling reviews of different laptops, the topic describing the preloaded software contains the words "software", "application", "programs", etc. in its shared distribution with high probability, and the Apple-specific word distribution contains the words "itunes", "appleworks", and "iphoto".

When generating a document under this model, one first samples a collection c (which is observable in the data), then chooses a topic z and flips a coin x to determine whether to draw from the shared topic-word distribution or the topic's collection-specific distribution. The probability of x being 1 or 0 comes from a Beta distribution (the bivariate analog of the Dirichlet distribution) and is dependent on the collection and topic of the current token.

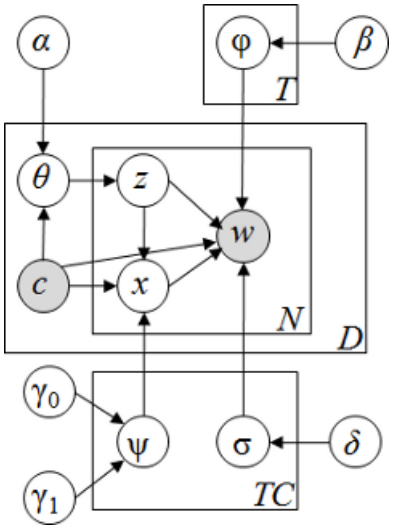


Figure 1: Graphical representation of ccLDA. C is the number of collections, T is the number of topics, D is the number of documents, and N is the length of each document.

The generative process is thus:

1) Draw a collection-independent multinomial word distribution ϕ_z from Dirichlet(β) for each topic z

2) Draw a collection-specific multinomial word distribution $\sigma_{z,c}$ from Dirichlet(δ) for each topic z and each collection c

3) Draw a Bernoulli distribution $\psi_{z,c}$ from Beta(γ_0, γ_1) for each topic z and each collection c

4) For each document d, choose a collection c and draw a topic mixture $\theta^{(d)}$ from Dirichlet(α_c). Then for each word w_i in d:

a) Sample a topic z_i from $\theta^{(d)}$

b) Sample x_i from $\psi_{z,c}$

c) If $x_i = 0$, sample a word w_i from ϕ_z;
else if $x_i = 1$, sample w_i from $\sigma_{z,c}$

As mentioned in section 2, this model is in some respects an LDA-based analog of the Zhai et al. (2004) model (ccMix), and thus it offers the same improvements that LDA offers over pLSI (described in the previous subsection), but there are some other differences. An obvious structural difference between the models is that ccMix has a special topic for background words, whereas we simply address this by removing stop words during preprocessing, which seems to give reasonable performance in this respect. This could easily be incorporated into our model such that x can take a

[1]One should also assume that a document length is sampled from an arbitrary distribution, but this does not affect the derivation of the model, so we ignore this here and elsewhere.

third value that designates that a word comes from the background, but removing stop words hugely reduces the number of tokens in the data, and thus very significantly improves the time needed to estimate the model.

In the ccMix model, the probability that a word comes from the collection-specific distribution versus the shared distribution depends on a single user-defined parameter λ_C. Since it is not clear how to set this parameter[2], in our model, we learn this probability automatically. Furthermore, the nature of the λ_C parameter is quite restrictive in that it is the same regardless of the topic and collection. In our model, this probability depends on the collection and topic, which should allow for a more accurate fitting of the data, as some topics may be shared across the collections to a different degree than others.

Additionally, our model allows the topic distributions for each document to come from non-uniform Dirichlet priors (parameterized by the vector α_c) that depends on the document's collection. Because the learned Dirichlet parameters can be interpreted as the average mixing level of each topic in the different collections, we can easily determine if a topic is not shared among all collections, and thus we can automatically remove or set aside such topics.

4 Parameter Estimation

Exact inference is often intractable in complex Bayesian models and approximate methods must be used. Blei et al. (2003) offer a variational EM algorithm for LDA. Griffiths and Steyvers (2004) show how Gibbs sampling can be used for approximate inference in LDA. Gibbs sampling is a type of Markov chain Monte Carlo algorithm and is what we employ in this paper, as it is simple to derive, comparable in speed to other estimators, and it approximates a global maximum (whereas EM algorithms may only converge to a local maximum).

In a Gibbs sampler, one iteratively samples new assignments of hidden variables by drawing from the distributions conditioned on the previous state of the model (Gilks et al., 1995). In each Gibbs sampling iteration we alternately sample new assignments of z and x with the following equations:

$$P(z_i|x_i = 0, \mathbf{z}_{-i}, \mathbf{w}, \alpha, \beta) \propto (n^d_{z_i} + \alpha_{cz}) \times \frac{n^{z_i}_{w_i} + \beta}{n^{z_i}_{\cdot} + W\beta} \quad (1)$$

$$P(z_i|x_i = 1, \mathbf{z}_{-i}, \mathbf{w}, \alpha, \delta) \propto (n^d_{z_i} + \alpha_{cz}) \times \frac{n^{z_i,c}_{w_i} + \delta}{n^{z_i,c}_{\cdot} + W\delta} \quad (2)$$

$$P(x_i = 0|\mathbf{x}_{-i}, \mathbf{z}, \mathbf{w}, \gamma, \beta) \propto \frac{n^{z,c}_{x=0} + \gamma_0}{n^{z,c}_{\cdot} + \gamma_0 + \gamma_1} \times \frac{n^{z_i}_{w_i} + \beta}{n^{z_i}_{\cdot} + W\beta} \quad (3)$$

$$P(x_i = 1|\mathbf{x}_{-i}, \mathbf{z}, \mathbf{w}, \gamma, \delta) \propto \frac{n^{z,c}_{x=1} + \gamma_1}{n^{z,c}_{\cdot} + \gamma_0 + \gamma_1} \times \frac{n^{z_i,c}_{w_i} + \delta}{n^{z_i,c}_{\cdot} + W\delta} \quad (4)$$

Because of the conjugacy of the Beta/Dirichlet and binomial/multinomial distributions, we can integrate out θ, ϕ, σ and ψ to obtain these equations, a technique known as "collapsed" Gibbs sampling (Heinrich, 2008).

n^b_a denotes the number of times a has been assigned to b, excluding the assignment of the current token i. W is the size of the vocabulary. x should be initialized as 0 for all tokens; that is, we initially assume that everything comes from the shared word distributions, otherwise the collection-specific word distributions will form independently.

α_c is a non-uniform vector that is collection-specific. A simple and efficient way to approximate this is through moment-matching such that $\alpha_{cz} \propto \frac{1}{N_c} \sum_d \frac{n^d_z}{n^d_\cdot}$, where d belongs to collection c and N_c is the number of documents in c (details in (Minka, 2003); (Li and McCallum, 2006)). The other hyperparameters can be updated similarly, although in our research we simply keep that at fixed, uniform values, as they do not largely affect the sampling procedure at small values.

5 Experimental Results

Our experiments focus on discovering cultural differences by running our model on text from or about three countries: the UK, India, and Singapore. We explore the notion of *perspective* by experimenting with datasets with two distinctly different perspectives: one in which the text is about each country (*tourists*), and one in which the text is authored by residents of each country (*locals*).

5.1 The Data

In our first experiment, we model 3,266 discussions from the forums at lonelyplanet.com, the largest blog website for travelers with a forum for nearly every potential travel destination. We show how this can be used for comparative content aggregation and summarization, and we show how

[2]If needed, one can effectively set this probability manually in ccLDA as well by using a large prior.

our model improves upon previous work on such datasets. In the second experiment, we compare by authorship (blogs written by *locals*), and we run our model on 7,388 English-language weblogs from the same set of three different countries[3]. We show how this is a solid step toward automatic discovery of cultural differences.

Moreover, we compare the two perspectives on the topic of food. We show that there are some strong similarities between the topic in each dataset (thereby enforcing our inferences from each experiment individually), but we also show some differences in the foods tourists find interesting and what locals actually eat.

In all of our experiments, we ran the Gibbs sampler for a burn-in period of 3000 iterations, then we collected and averaged 15 samples, each separated by a 100-iteration lag. We used $\beta = \delta = 0.01$ and $\gamma_0 = \gamma_1 = 1.0$.

Our implementation is loosely based on the LDA Gibbs sampler[4] by Phan and Nguyen (2008).

5.2 Analysis Along the *Tourists* Dimension

In the first experiment we consider data about three destination countries. Using the data provided by lonelyplanet.com, we crawled 1,108 threads from the UK forum, 1,112 from the India forum, and 1,046 from the Singapore forum. Messages are predominantly written by people who have traveled or plan to travel to that country.

Since we are not interested in the thread discussions on a particular travel topic, we treated each thread or discussion of multiple messages as a single document. We were able to use simple pattern matching to extract only the discussion text. We removed HTML tags, stop words, and words with a corpus frequency less than 10. There were 703,551 tokens after preprocessing.

We modeled this dataset with 25 topics. General topical words were grouped into the shared word distribution of each topic, but each collection-specific distribution contained words in the topic that best describe that country. For example, the topic on weather is characterized by words like *weather*, *rain* and *snow*, but each collection's distribution might give one a sense of the weather in each country. Table 1 shows that travelers in India, for example, should be aware of monsoon season, and travelers to Singapore can expect to be hot and sweaty. The UK distribution suggests that campers should prepare for potentially hazardous weather with the appropriate clothing and gear.

weather time day going rain summer month high days thanks		
UK	**India**	**Singapore**
wind	leh	hot
waterproof	monsoon	humid
ending	road	humidity
rolling	manali	heat
walkers	ladakh	degree
rochdale	trekking	equator
layers	trek	sweat
snow	season	bring
footwear	rains	rain
ankle	monsoons	umbrella

Table 1: The topic of weather, modeled across travel forums for three different countries.

As another example, let's consider the topic whose shared words are *english*, *school*, *language*, and *speak*. The results show that English is common to all three, but the collection-specific word distributions indicate that Irish language is found in the UK region, Hindi is common in India, and Mandarin is common in Singapore.

Other common topics include immigration requirements, monetary issues, air and rail travel, etc., all containing information specific to each country. This could be used for automatic summarization by topic which would be useful either to travelers who are visiting multiple destinations, or for a potential traveler in the process of choosing where to go. Someone interested in shopping for music should go to the UK while someone interested in electronics should go to Singapore, for example (at least according to one of the topics discovered).

5.3 Analysis Along the *Locals* Dimension

The results of the first experiment offer an unsupervised aggregation of factual information that is important to travelers such as a destination's climate, law, and infrastructure; however, the data did not offer much in terms of cultural information. We would now like to see if we can get better insight into this problem by modeling text *authored* by residents of these same countries. In doing this we can compare what they talk about and in what manner they talk about certain topics.

For this experiment we downloaded 2,715 blogs from the UK, 2,630 blogs from India, and 2,043 blogs from Singapore. We found these English-language blogs through blogcatalog.com, a blog

[3]The dataset is available for download at http://apfel.ai.uiuc.edu/resources.html

[4]http://gibbslda.sourceforge.net

directory which lists a blog's language and country of origin. We downloaded only the front page of each blog, which usually included multiple articles or postings.

We removed HTML tags from the documents, but we made no attempt to segment the documents into article text – there are efficient methods of doing this (Pasternack and Roth, 2009) and this may be worth experimenting with, but we found that noise such as navigation menus and advertisements would mostly get grouped into their own topics. We removed stop words and words with a corpus frequency less than 20. All punctuation was treated as word separators. There were 8,599,751 tokens in the end.

Table 2 shows 3 topics induced from modeling this data with 50 topics. By looking at these we can see some clear differences between the three groups of native bloggers. For example, Topic 1 is about fashion, and we can compare which fashions are popular in each country. Shoes are popular in the UK; leather and jewelry are more popular in India. Singapore bloggers seem to focus on prices and the shopping aspect of apparel.

From Topic 2 (about pets) it seems that Britons slightly prefer dogs and Singaporians slightly prefer cats. In general, it seems that Singaporians have an affinity for small animals, considering the presence of *hamster* and *rabbit* in their word distribution.

Topic 3 is about religion, in which we see that Christianity is common to all of them, but Hinduism is prominent in India as well.

There are many topics not shown here including politics, gardening, health, etc. The health topic is interesting in that homeopathy and herbal medicines are discussed in Indian blogs. Smoking is a bigger topic in the UK than the others.

It is also interesting to compare what technologies and web services people use. Twitter and Facebook are popular in the UK whereas Orkut is more popular in India. Blogging services like Wordpress are popular in Singapore.

From the travel topic, shown in Table 5, we see that people travel close to home, so to speak. Britons travel around Europe, especially Spain, Paris and London, while Singaporians travel to popular destinations in that part of the world, such as Hong Kong, Thailand and Bali.

5.4 Differences in Perspective: *Tourists* vs. *Locals*

Having modeled the same countries from two different perspectives (that of travelers and that of locals), it would be interesting to see how topics compare between the two perspectives.

Do people have the same view of themselves as outsiders see them? Are locals interested in the same things as tourists?

We hope to answer these questions by examining related topics within these two datasets. While the two datasets consist of mostly different topics, there are a few that would be interesting to compare. In particular, we examine the topic of food and eating. The top words from this topic are shown in Table 3.

We first examine this topic from the blog data (that is, from the perspective of residents). By looking at each collection-specific word distribution we can see which foods are more popular in each country – cheese and soup in the UK, curry in India, and seafood in Singapore. We also noticed that tea and coffee are more popular in Singapore, wine and beer are more popular in the UK, while in Indian blogs beverages are not commonly mentioned. Perhaps a less trivial observation is that the words *restaurant* and *chef* are frequent in UK blogs, but the Indian word distribution is dominated by words pertaining to recipes. From this one might infer that people in the UK (and to a lesser extent in Singapore) eat out more often than people in India, who do more home cooking.

Looking now at the topics induced from the lonelyplanet.com forums (that is, from the perspective of travelers), we see some interesting similarities. Most notably, the Indian distribution again consists of words related to cooking, affirming our observation that dining out is not as popular in India. The Singapore distribution also matches that in the other dataset – the common words include seafood and noodles. The UK distribution, however, shows that tourists are mostly interested in local specialties (such as *fish and chips* and *haggis*).

To see where these perspectives on food differ the most, we computed the ratio of the probability of each word given the topic between the two datasets. That is, if $p = P(w|z)$ in the locals data and $q = P(w|z)$ in the tourists data, then $\lambda = p/q$ gives us a measure of how much more (or less) prominent that word is among locals than it

Topic 1			Topic 2			Topic 3		
fashion style look dress wear new collection accessories black			dog dogs pet animals animal comments cat like food plant			god jesus lord life faith holy man christ church love		
UK	**India**	**Singapore**	**UK**	**India**	**Singapore**	**UK**	**India**	**Singapore**
shoes	fashion	price	garden	water	cat	church	krishna	god
fashion	women	posted	dog	energy	cats	god	religion	sin
clothing	indian	earrings	pet	carbon	dog	john	religious	john
high	designer	length	cat	earth	pet	todd	spiritual	spirit
designer	sarees	item	dogs	green	training	bentley	guru	things
style	leather	sgd	pets	solar	pets	jesus	lord	lamb
love	girls	silver	gardening	jai	hamster	christ	sri	exodus
london	china	clothes	cats	climate	cute	luke	shri	suffering
shirts	jewellery	shop	puppy	environment	hamsters	bible	baba	cross
bag	jewelry	code	flowers	warming	rabbit	christian	hindu	lives

Table 2: A sample of topics induced on a set of blogs from 3 countries. Shown are the top 10 words from the shared topic-word distribution $P(word|x = 0, topic)$ and the top 10 words from $P(word|x = 1, topic, class)$ for each collection.

Perspective of Locals			Perspective of Tourists		
food add chicken recipe cooking taste rice recipes sugar soup			food eat restaurant restaurants tea cheap meal eating cafe drink		
UK	**India**	**Singapore**	**UK**	**India**	**Singapore**
food	recipe	coffee	chips	cooking	hawker[a]
wine	recipes	cup	haggis	spices	satay
restaurant	powder	oil	fish	sick	stalls
coffee	indian	comments	respectability	flour	noodles
cheese	salt	fried	decent	tomato	roti
soup	tsp	add	veggie	batter	stall
eat	rice	restaurant	pudding	ate	seafood
chef	masala	rice	photoblog	cook	malay
english	oil	tea	sausages	olive	rochester
drink	coriander	seafood	sandwiches	recipe	noodle

[a]A hawker centre is an open-air complex with many food stalls, commonly found in Singapore and Malaysia.

Table 3: A comparison of the food topic from two different datasets, one of which comes from a travel forum and the other of which consists of blogs authored by residents of each respective country.

is among tourists in the food topic. Table 4 shows the words with the highest (left) and lowest (right) values of λ.

Preferred by Locals			Preferred by Tourists		
recipe bowl lemon tomato simple spring spoon vanilla stir pour			street cheap couple yeah crowd old road floor run locals		
UK	**India**	**Singapore**	**UK**	**India**	**Singapore**
food	indian	cup	pubs	mother	quay
healthy	recipes	comments	music	ate	coast
shop	cup	tea	lane	tree	parkway
favorite	chicken	mins	brick	party	reasonably
wine	minutes	pot	fish	fields	air
icing	kitchen	note	jazz	base	sultan
coffee	mustard	nice	pints	rock	tum
leeds	fried	salt	dancing	toilet	views
duck	ginger	tarts	arms	bottled	plenty
extra	salt	fish	recommend	olive	rochester

Table 4: This table shows words in the food topic that are more popular in the tourists data than the locals data or vice versa.

The prominent trend, which is largely a logistical matter, is that travelers are more interested in restaurants and locals talk more about cooking. Most of the words that are more prominent from the tourist perspective have to do with eating locations. We also noticed that wine and coffee rank more prominently among the locals, whereas travelers are more likely to ask about beer and liquor.

5.5 Model Evaluation

In this subsection we evaluate ccLDA against ccMix and LDA both qualitatively, through blind judgments of cluster quality, and quantitatively, by measuring the likelihood of held-out data with each model.

5.5.1 Cluster Coherence

Because our research relies on analyses of discovered topics, it is important that we use a model that gives the best empirical quality of word clusters. We compare against ccMix (Zhai et al., 2004), the only related model that is naturally suited to our task. Using blind human judgments we show that ccLDA unquestionably delivers topics that are more coherent than those obtained with the ccMix model.

A direct comparison with ccMix is tricky because it incorporates a model for background words, whereas our model expects stop words to be removed during preprocessing. So that they are fully comparable, we set the parameter λ_B (the probability that a word comes from the background) to 0 and fed the model the same input as we did ccLDA. We set the parameter λ_C, analogous to $P(x = 0)$, to 0.6, which is the average value learned by ccLDA on this data, and it seems quite reasonable. Using an implementation provided by the authors of ccMix, we ran the EM procedure for 20 trials and saved the model with the best log-likelihood.

We performed human judgments of the 25 topics induced by ccLDA in the first experiment

above and by the ccMix model with the number of topics set again to 25. We aligned the topics automatically using a symmetric KL-divergence score computed on the collection-independent distributions – specifically, $D(P||Q) + D(Q||P)$ where $D(P||Q)$ is the KL-divergence[5] of the distributions P and Q.

Each aligned pair of topics (ordered randomly for each topic to avoid bias) was presented to two natural language processing researchers who were asked to choose which one was better, based on the following criteria: (1) semantic coherence of the topic as a whole (e.g. are the words in the clusters related?) and (2) coherence across collections, that is, are the collection-specific distributions related to each other and to the common one? The judges were also given the option to rate a pair as "no opinion" in the case that the aligned topics were too dissimilar to compare (because the two models did not discover the same topic), or that the topics did not carry enough semantic information to judge (i.e. topics composed mostly of function words).

Of the 25 pairs, there were 10 that both judges rated. Of these 10, the judges disagreed on 3. The other 7 were all rated in favor of ccLDA.

Similarly, the 50 topics from the second experiment were judged against 50 topics formed using ccMix. There were 22 topics that both judges rated. Among these, they disagreed on only 3; of the remaining topics they voted in favor of ccMix for 1 topic and in favor of ccLDA for 18 topics.

It has been observed that the performance of a model can largely depend on the estimator used (Girolami and Kabán, 2003), so it may be that the weaker performance of ccMix is because the EM algorithm is getting stuck in local maxima, even after several trials.

Table 5 shows the topic of travel compared with both ccMix and LDA. To compare against LDA, we performed a post-hoc estimation of the topic's word distribution for each collection by considering topic assignments of documents within each collection. We see that the ccLDA distributions are much more coherent than that of ccMix. Furthermore, the advantage over LDA is clear – with LDA, we do not get a separation of the words that are common to all of the collections, and thus it is hard to detect the important differences at a glance.

[5]Kullback-Leibler divergence is a commonly used measurement of the similarity of two probability distributions.

5.5.2 Likelihood Comparison

To measure how well our model can generalize unseen documents, we compute the likelihood of held-out data using ccLDA compared with ccMix and LDA. We partitioned the forum dataset from the first experiment into a subset of 80% of the data on which the models are learned, and an evaluation set of the remaining 20%.

To calculate the likelihood of the held-out documents with ccMix, we use the "fold-in" method (Hofmann, 1999) in which the mixing proportions except for $P(z|d)$ are fixed during the EM process. As with our cluster evaluation above, we set $\lambda_B = 0$ and $\lambda_C = 0.6$. With LDA and ccLDA, we approximate $P(z|d)$ through another Gibbs sampling procedure, by averaging 10 samples collected after 100 iterations with a 10-iteration lag in between each sample.

The log-likelihood of the three models is shown at various numbers of topics in Figure 2. As expected, ccLDA generally achieves a higher likelihood than ccMix, although the difference between them diminishes at higher numbers of topics. This appears to be because the pLSI-based ccMix does not regularize the topic mixtures and can thus achieve higher values of $P(z|d)$, and the smoothing of ccLDA has a greater effect at higher numbers of topics.

Both cross-collection models achieve a higher likelihood than LDA, which is not too surprising, given that these models utilize extra information (specifically, the document's collection) to assign a higher probability to words more likely to appear in a document given that information.

It should be noted that even though the likelihood of both cross-collection models increases with the number of topics up to 100, we observed empirically that the best cluster quality in this dataset occurs around 20 to 30 topics; more than that results in clusters that are repeated and are largely specific to only one collection.

6 Discussion and Future Work

While there are obvious limitations of the unigram approach used here, our system was nevertheless able to capture some interesting details. It is important, however, to point out some limitations for possible future extensions.

Consider Topic 2 in Table 2. The UK and Singapore word distributions are both clearly pertinent

ccLDA			ccMix			LDA		
travel hotel hotels city best place holiday visit trip world			travel hotel comments hotels city posted road trip labels airport			travel city hotel park holiday hotels place beach road visit		
UK	**India**	**Singapore**	**UK**	**India**	**Singapore**	**UK**	**India**	**Singapore**
holiday	india	singapore	yang	india	yang	travel	travel	travel
holidays	delhi	kong	train	delhi	dan	holiday	city	hotel
hotels	indian	hong	london	tourism	ini	hotel	beach	city
spain	mumbai	spa	saya	dubai	dengan	city	place	park
london	bangalore	hotel	nie	indian	untuk	london	hotel	place
great	tour	beach	travel	tour	itu	park	temple	beach
surf	air	chinese	flight	bangalore	saya	hotel	road	trip
breaks	dubai	pictures	luxury	mahindra	orang	place	park	hotels
train	city	restaurant	dan	hotels	tidak	holidays	hotels	spa
ski	mahindra	bangkok	advert	marathi	dalam	hall	tourism	visit

Table 5: The topic of *travel* as discovered by the 3 different models.

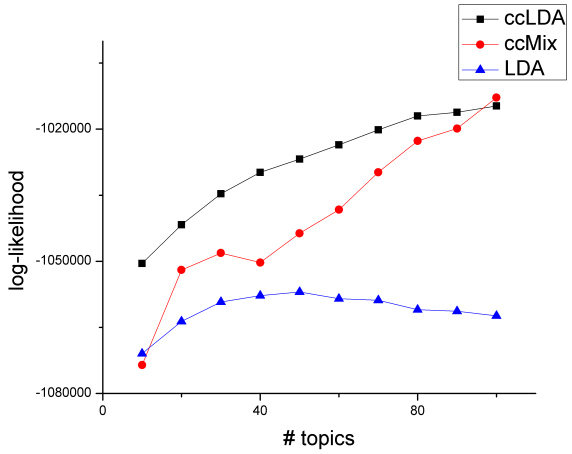


Figure 2: Comparison of the log-likelihood of held-out data with the 3 models.

to the topic of pets, but the India distribution seems entirely unrelated, being about energy and the environment. This could be because the environment topic was statistically too strong to ignore, but not found in other collections, so it made its way into a largely unrelated topic. (In fact, the formation of the environment cluster within this topic is not entirely random, as the pets topic also includes some words related to gardening, including "water" and "plant", which are likely to also co-occur with environmental words.)

This is perhaps the main weakness of the model. If an emerging topic is not shared among all collections, it will either form as a primary topic that is unique to only a subset of collections (and thus some of the collection-specific distributions will be noisy), or it will form as a collection-specific distribution that is not strongly related to the main collection-independent distribution. This can make the results difficult to interpret, although an automated solution would be to remove or flag topics that are not evenly shared, which could be done by comparing the learned collection-dependent Dirichlet parameters α_c.

This is also a matter of how the model performs with different numbers of collections. It would be interesting to see what results we would get by modeling UK-India, UK-Singapore, and India-Singapore as only a pair at a time. The performance should not degrade with larger numbers of collections if the collections are fully comparable, but in practice, with more collections there are likely to be more topics that are difficult to fit across all collections.

In future work, we would like to enrich the model and/or feature set to move beyond the limitations of a bag-of-words analysis. For example, by considering negation and word polarity, we can better capture the opinions of the authors, which is an important component of such cultural analysis.

Certainly, there are many other possible applications of this model, including product comparison, media bias detection, and interdisciplinary literature analysis. Cultural awareness is also important in marketing and we can use this model to investigate, for example what products and what aspects of life people in different regions focus on.

Acknowledgments

We would like to thank ChengXiang Zhai for thoughtful discussions and for providing an implementation of the ccMix model. We would also like to thank the reviewers for their constructive comments.

References

D. Blei, A. Ng, and M. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3.

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

W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. 1995. *Markov Chain Monte Carlo in Practice*. CRC Press.

M. Girolami and A. Kabán. 2003. On an equivalence between plsi and lda. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 433–434, New York, NY, USA. ACM.

T. Griffiths and M. Steyvers. 2004. Finding scientific topics. In *Proceedings of the National Academy of Sciences of the United States of America*.

Tl Griffiths, M. Steyvers, and Jb Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114(2):211–244.

D. Hall, D. Jurafsky, and C. Manning. 2008. Studying the history of ideas using topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 363–371.

G. Heinrich. 2008. Parameter estimation for text analysis. Technical report, University of Leipzig.

T. Hofmann. 1999. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, New York, NY, USA. ACM.

W. Li and A. McCallum. 2006. Pachinko allocation: Dag-structured mixture models of topic correlations. In *International Conference on Machine Learning*.

A. McCallum, X. Wang, and A. Corrada-Emmanuel. 2007a. Topic and role discovery in social networks with experiments on enron and academic email. *Journal of Artificial Intelligence Research (JAIR)*, 30:249–272.

A. McCallum, X. Wang, and N. Mohanty. 2007b. Joint group and topic discovery from relations and text. In *Statistical Network Analysis: Models, Issues and New Directions - Lecture Notes in Computer Science 4503*, pages 28–44.

D. Mimno, W. Li, and A. McCallum. 2007. Mixtures of hierarchical topics with pachinko allocation. In *International Conference on Machine Learning*.

T. Minka. 2003. Estimating a dirichlet distribution.

T. Mitchell. 1997. *Machine Learning*. McGraw-Hill, Boston.

J. Pasternack and D. Roth. 2009. Extracting article text from the web with maximum subsequence segmentation. In *The International World Wide Web Conference*, April.

M. Paul and R. Girju. 2009. Topic modeling of research fields: An interdisciplinary perspective. In *Proceedings of the the International Conference on Recent Advances in Natural Language Processing (RANLP) (to appear)*.

X. Phan, L. Nguyen, and S. Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 91–100, New York, NY, USA. ACM.

X. Wang, A. McCallum, and X. Wei. 2007. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 697–702. IEEE Computer Society.

C. Wang, B. Thiesson, C. Meek, and D. Blei. 2009. Markov topic models. In *The Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 583–590.

T. Yano, W. Cohen, and N. Smith. 2009. Predicting response to political blog posts with topic models. In *The 7th Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

C. Zhai, A. Velivelli, and B. Yu. 2004. A cross-collection mixture model for comparative text mining. In *Proceedings of KDD 04*, pages 743–748.

Consensus Training for Consensus Decoding in Machine Translation

Adam Pauls, John DeNero and **Dan Klein**
Computer Science Division
University of California at Berkeley
{adpauls,denero,klein}@cs.berkeley.edu

Abstract

We propose a novel objective function for discriminatively tuning log-linear machine translation models. Our objective explicitly optimizes the BLEU score of *expected* n-gram counts, the same quantities that arise in forest-based consensus and minimum Bayes risk decoding methods. Our continuous objective can be optimized using simple gradient ascent. However, computing critical quantities in the gradient necessitates a novel dynamic program, which we also present here. Assuming BLEU as an evaluation measure, our objective function has two principle advantages over standard max BLEU tuning. First, it specifically optimizes model weights for downstream consensus decoding procedures. An unexpected second benefit is that it reduces overfitting, which can improve test set BLEU scores when using standard Viterbi decoding.

1 Introduction

Increasing evidence suggests that machine translation decoders should not search for a single top scoring Viterbi derivation, but should instead choose a translation that is sensitive to the model's entire predictive distribution. Several recent consensus decoding methods leverage compact representations of this distribution by choosing translations according to n-gram posteriors and expected counts (Tromble et al., 2008; DeNero et al., 2009; Li et al., 2009; Kumar et al., 2009). This change in decoding objective suggests a complementary change in *tuning* objective, to one that optimizes expected n-gram counts directly. The ubiquitous minimum error rate training (MERT) approach optimizes Viterbi predictions, but does not explicitly boost the aggregated posterior probability of desirable n-grams (Och, 2003).

We therefore propose an alternative objective function for parameter tuning, which we call *consensus* BLEU or CoBLEU, that is designed to maximize the *expected counts* of the n-grams that appear in reference translations. To maintain consistency across the translation pipeline, we formulate CoBLEU to share the functional form of BLEU used for evaluation. As a result, CoBLEU optimizes exactly the quantities that drive efficient consensus decoding techniques and precisely mirrors the objective used for fast consensus decoding in DeNero et al. (2009).

CoBLEU is a continuous and (mostly) differentiable function that we optimize using gradient ascent. We show that this function and its gradient are efficiently computable over packed forests of translations generated by machine translation systems. The gradient includes expectations of *products* of features and n-gram counts, a quantity that has not appeared in previous work. We present a new dynamic program which allows the efficient computation of these quantities over translation forests. The resulting gradient ascent procedure does not require any k-best approximations. Optimizing over translation forests gives similar stability benefits to recent work on lattice-based minimum error rate training (Macherey et al., 2008) and large-margin training (Chiang et al., 2008).

We developed CoBLEU primarily to complement consensus decoding, which it does; it produces higher BLEU scores than coupling MERT with consensus decoding. However, we found an additional empirical benefit: CoBLEU is less prone to overfitting than MERT, even when using Viterbi decoding. In experiments, models trained to maximize tuning set BLEU using MERT consistently degraded in performance from tuning to test set, while CoBLEU-trained models generalized more robustly. As a result, we found that optimizing CoBLEU improved test set performance reliably using consensus decoding and occasionally using Viterbi decoding.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1418–1427,
Singapore, 6-7 August 2009.

(a) **Hypotheses ranked by $\theta_{TM} = \theta_{LM} = 1$**

Sentence *f*: Il était une rime
Reference *r*: Once upon a rhyme

	TM	LM	Pr
H_1) Once on a rhyme	-3	-7	0.67
H_2) Once upon a rhyme	-5	-6	0.24
H_3) Once upon a time	-9	-3	0.09

(b) **Computing Consensus Bigram Precision**

$$\begin{aligned}
\mathbb{E}_\theta[c(\text{“Once upon”}, d)|f] &= 0.24 + 0.09 = 0.33 \\
\mathbb{E}_\theta[c(\text{“upon a”}, d)|f] &= 0.24 + 0.09 = 0.33 \\
\mathbb{E}_\theta[c(\text{“a rhyme”}, d)|f] &= 0.67 + 0.24 = 0.91 \\
\sum_g \mathbb{E}_\theta[c(g, d)|f] &= 3[0.67 + 0.24 + 0.09] \\
\frac{\sum_g \min\{\mathbb{E}_\theta[c(g, d)|f], c(g, r)\}}{\sum_g \mathbb{E}_\theta[c(g, d)|f]} &= \frac{0.33 + 0.33 + 0.91}{3}
\end{aligned}$$

Figure 1: (a) A simple hypothesis space of translations for a single sentence containing three alternatives, each with two features. The hypotheses are scored under a log-linear model with parameters θ equal to the identity vector. (b) The expected counts of all bigrams that appear in the computation of consensus bigram precision.

2 Consensus Objective Functions

Our proposed objective function maximizes n-gram precision by adapting the BLEU evaluation metric as a tuning objective (Papineni et al., 2002). To simplify exposition, we begin by adapting a simpler metric: bigram precision.

2.1 Bigram Precision Tuning

Let the tuning corpus consist of source sentences $F = f_1 \dots f_m$ and human-generated references $R = r_1 \dots r_m$, one reference for each source sentence. Let e_i be a translation of f_i, and let $E = e_1 \dots e_m$ be a corpus of translations, one for each source sentence. A simple evaluation score for E is its bigram precision $\text{BP}(R, E)$:

$$\text{BP}(R, E) = \frac{\sum_{i=1}^m \sum_{g_2} min\{c(g_2, e_i), c(g_2, r_i)\}}{\sum_{i=1}^m \sum_{g_2} c(g_2, e_i)}$$

where g_2 iterates over the set of bigrams in the target language, and $c(g_2, e)$ is the count of bigram g_2 in translation e. As in BLEU, we “clip” the bigram counts of e in the numerator using counts of bigrams in the reference sentence.

Modern machine translation systems are typically tuned to maximize the evaluation score of Viterbi derivations[1] under a log-linear model with parameters θ. Let $d^*_\theta(f_i) = \arg\max_d P_\theta(d|f_i)$ be the highest scoring derivation d of f_i. For a system employing Viterbi decoding and evaluated by bigram precision, we would want to select θ to maximize $\text{MaxBP}(R, F, \theta)$:

$$\frac{\sum_{i=1}^m \sum_{g_2} min\{c(g_2, d^*_\theta(f_i)), c(g_2, r_i)\}}{\sum_{i=1}^m \sum_{g_2} c(g_2, d^*_\theta(f_i))}$$

On the other hand, for a system that uses expected bigram counts for decoding, we would prefer to choose θ such that expected bigram counts match bigrams in the reference sentence. To this end, we can evaluate an entire posterior distribution over derivations by computing the same clipped precision for expected bigram counts using $\text{CoBP}(R, F, \theta)$:

$$\frac{\sum_{i=1}^m \sum_{g_2} min\{\mathbb{E}_\theta[c(g_2, d)|f_i], c(g_2, r_i)\}}{\sum_{i=1}^m \sum_{g_2} \mathbb{E}_\theta[c(g_2, d)|f_i]} \quad (1)$$

where

$$\mathbb{E}_\theta[c(g_2, d)|f_i] = \sum_d P_\theta(d|f_i) c(g_2, d)$$

is the expected count of bigram g_2 in all derivations d of f_i. We define the precise parametric form of $P_\theta(d|f_i)$ in Section 3. Figure 1 shows proposed translations for a single sentence along with the bigram expectations needed to compute CoBP.

Equation 1 constitutes an objective function for tuning the parameters of a machine translation model. Figure 2 contrasts the properties of CoBP and MaxBP as tuning objectives, using the simple example from Figure 1.

Consensus bigram precision is an instance of a general recipe for converting n-gram based evaluation metrics into consensus objective functions for model tuning. For the remainder of this paper, we focus on consensus BLEU. However, the techniques herein, including the optimization approach of Section 3, are applicable to many differentiable functions of expected n-gram counts.

[1] By *derivation*, we mean a translation of a foreign sentence along with any latent structure assumed by the model. Each derivation corresponds to a particular English translation, but many derivations may yield the same translation.

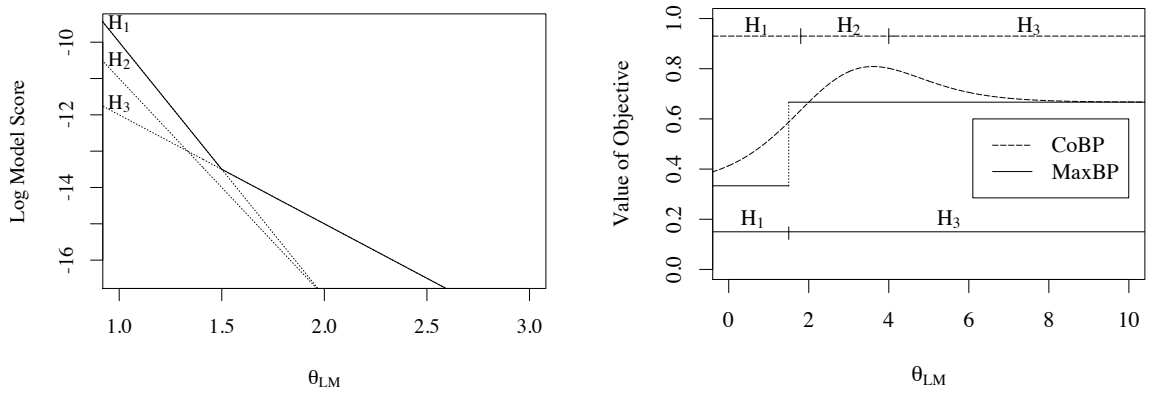


Figure 2: These plots illustrate two properties of the objectives *max bigram precision* (MaxBP) and *consensus bigram precision* (CoBP) on the simple example from Figure 1. (a) MaxBP is only sensitive to the convex hull (the solid line) of model scores. When varying the single parameter θ_{LM}, it entirely disregards the correct translation H_2 because H_2 never attains a maximal model score. (b) A plot of both objectives shows their differing characteristics. The horizontal segmented line at the top of the plot indicates the range over which consensus decoding would select each hypothesis, while the segmented line at the bottom indicates the same for Viterbi decoding. MaxBP is only sensitive to the single point of discontinuity between H_1 and H_3, and disregards H_2 entirely. CoBP peaks when the distribution most heavily favors H_2 while suppressing H_1. Though H_2 never has a maximal model score, if θ_{LM} is in the indicated range, consensus decoding would select H_2, the desired translation.

2.2 CoBLEU

The logarithm of the single-reference[2] BLEU metric (Papineni et al., 2002) has the following form:

$$\ln \text{BLEU}(R, E) = \left(1 - \frac{|R|}{\sum_{i=1}^{m} \sum_{g_1} c(g_1, e_i)}\right)_- + \frac{1}{4} \sum_{n=1}^{4} \ln \frac{\sum_{i=1}^{m} \sum_{g_n} \min\{c(g_n, e_i), c(g_n, r_i)\}}{\sum_{i=1}^{m} \sum_{g_n} c(g_n, e_i)}$$

Above, $|R|$ denotes the number of words in the reference corpus. The notation $(\cdot)_-$ is shorthand for $min(\cdot, 0)$. In the inner sums, g_n iterates over all n-grams of order n. In order to adapt BLEU to be a consensus tuning objective, we follow the recipe of Section 2.1: we replace n-gram counts from a candidate translation with expected n-gram counts under the model.

$$\text{CoBLEU}(R, F, \theta) = \left(1 - \frac{|R|}{\sum_{i=1}^{m} \sum_{g_1} \mathbb{E}_\theta[c(g_1, d)|f_i]}\right)_- + \frac{1}{4} \sum_{n=1}^{4} \ln \frac{\sum_{i=1}^{m} \sum_{g_n} \min\{\mathbb{E}_\theta[c(g_n, d)|f_i], c(g_n, r_i)\}}{\sum_{i=1}^{m} \sum_{g_n} \mathbb{E}_\theta[c(g_n, d)|f_i]}$$

The brevity penalty term in BLEU is calculated using the expected length of the corpus, which equals the sum of all expected unigram counts. We call this objective function *consensus* BLEU, or CoBLEU for short.

3 Optimizing CoBLEU

Unlike the more common MaxBLEU tuning objective optimized by MERT, CoBLEU is continuous. For distributions $P_\theta(d|f_i)$ that factor over synchronous grammar rules and n-grams, we show below that it is also analytically differentiable, permitting a straightforward gradient ascent optimization procedure.[3] In order to perform gradient ascent, we require methods for efficiently computing the gradient of the objective function for a given parameter setting θ. Once we have the gradient, we can perform an update at iteration t of the form

$$\theta^{(t+1)} \leftarrow \theta^{(t)} + \eta_t \nabla_\theta \text{CoBLEU}(R, F, \theta^{(t)})$$

where η_t is an adaptive step size.[4]

[2]Throughout this paper, we use only a single reference, but our objective readily extends to multiple references.

[3]Technically, CoBLEU is non-differentiable at some points because of *clipping*. At these points, we must compute a sub-gradient, and so our optimization is formally sub-gradient ascent. See the Appendix for details.

[4]After each successful step, we grow the step size by a constant factor. Whenever the objective does not decrease after a step, we shrink the step size by a constant factor and try again until a decrease is attained.

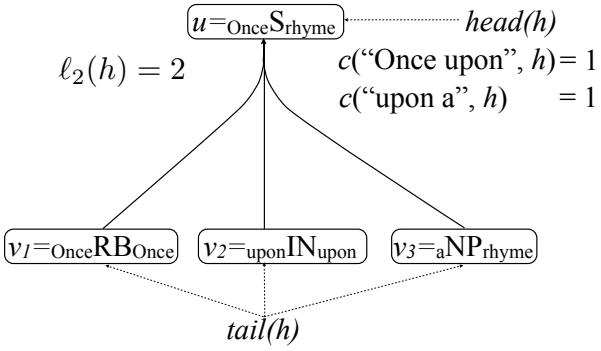


Figure 3: A hyperedge h represents a "rule" used in syntactic machine translation. *tail(h)* refers to the "children" of the rule, while $head(h)$ refers to the "head" or "parent". A forest of translations is built by combining the nodes v_i using h to form a new node $u = head(h)$. Each forest node consists of a grammar symbol and target language boundary words used to track n-grams. In the above, we keep one boundary word for each node, which allows us to track bigrams.

In this section, we develop an analytical expression for the gradient of CoBLEU, then discuss how to efficiently compute the value of the objective function and gradient.

3.1 Translation Model Form

We first assume the general hypergraph setting of Huang and Chiang (2007), namely, that derivations under our translation model form a hypergraph. This framework allows us to speak about both phrase-based and syntax-based translation in a unified framework.

We define a probability distribution over derivations d via θ as:

$$P_\theta(d|f_i) = \frac{w(d)}{Z(f_i)}$$

with

$$Z(f_i) = \sum_{d'} w(d')$$

where $w(d) = \exp(\theta^\top \Phi(d, f_i))$ is the weight of a derivation and $\Phi(d, f_i)$ is a featurized representation of the derivation d of f_i. We further assume that these features decompose over *hyperedges* in the hypergraph, like the one in Figure 3. That is, $\Phi(d, f_i) = \sum_{h \in d} \Phi(h, f_i)$.

In this setting, we can analytically compute the gradient of CoBLEU. We provide a sketch of the derivation of this gradient in the Appendix. In computing this gradient, we must calculate the following expectations:

$$\mathbb{E}_\theta\left[c(\phi_k, d)|f_i\right] \quad (2)$$

$$\mathbb{E}_\theta\left[\ell_n(d)|f_i\right] \quad (3)$$

$$\mathbb{E}_\theta\left[c(\phi_k, d) \cdot \ell_n(d)|f_i\right] \quad (4)$$

where $\ell_n(d) = \sum_{g_n} c(g_n, d)$ is the sum of all n-grams on derivation d (its "length"). The first expectation is an expected count of the kth feature ϕ_k over all derivations of f_i. The second is an expected length, the total expected count of all n-grams in derivations of f_i. We call the final expectation an expected *product of counts*. We now present the computation of each of these expectations in turn.

3.2 Computing Feature Expectations

The expected feature counts $\mathbb{E}_\theta[c(\phi_k, d)|f_i]$ can be written as

$$\begin{aligned}\mathbb{E}_\theta[c(\phi_k, d)|f_i] &= \sum_d P_\theta(d|f_i)c(\phi_k, d) \\ &= \sum_h P_\theta(h|f_i)c(\phi_k, h)\end{aligned}$$

We can justify the second step since feature counts are local to hyperedges, i.e. $c(\phi_k, d) = \sum_{h \in d} c(\phi_k, h)$. The posterior probability $P_\theta(h|f_i)$ can be efficiently computed with inside-outside scores. Let $\mathrm{I}(u)$ and $\mathrm{O}(u)$ be the standard inside and outside scores for a node u in the forest.[5]

$$P_\theta(h|f_i) = \frac{1}{Z(f)}\, w(h)\, \mathrm{O}(head(h)) \prod_{v \in tail(h)} \mathrm{I}(v)$$

where $w(h)$ is the weight of hyperedge h, given by $exp(\theta^\top \Phi(h))$, and $Z(f) = \mathrm{I}(root)$ is the inside score of the root of the forest. Computing these inside-outside quantities takes time linear in the number of hyperedges in the forest.

3.3 Computing n-gram Expectations

We can compute the expectations of any specific n-grams, or of total n-gram counts ℓ, in the same way as feature expectations, provided that target-side n-grams are also localized to hyperedges (e.g. consider ℓ to be a feature of a hyperedge whose value is the number of n-grams on h). If the nodes in our forests are annotated with target-side

[5] Appendix Figure 7 gives recursions for $\mathrm{I}(u)$ and $\mathrm{O}(u)$.

boundary words as in Figure 3, then this will be the case. Note that this is the same approach used by decoders which integrate a target language model (e.g. Chiang (2007)). Other work has computed n-gram expectations in the same way (DeNero et al., 2009; Li et al., 2009).

3.4 Computing Expectations of Products of Counts

While the previous two expectations can be computed using techniques known in the literature, the expected product of counts $\mathbb{E}_\theta[c(\phi_k, d) \cdot \ell_n(d)|f_i]$ is a novel quantity. Fortunately, an efficient dynamic program exists for computing this expectation as well. We present this dynamic program here as one of the contributions of this paper, though we omit a full derivation due to space restrictions.

To see why this expectation cannot be computed in the same way as the expected feature or n-gram counts, we expand the definition of the expectation above to get

$$\sum_d P_\theta(d|f_i)\,[c(\phi_k, d)\ell_n(d)]$$

Unlike feature and n-gram counts, the product of counts in brackets above does not decompose over hyperedges, at least not in an obvious way. We can, however, still decompose the feature counts $c(\phi_k, d)$ over hyperedges. After this decomposition and a little re-arranging, we get

$$\begin{aligned}
&= \sum_h c(\phi_k, h) \sum_{d:h\in d} P_\theta(d|f_i)\ell_n(d) \\
&= \frac{1}{Z(f_i)} \sum_h c(\phi_k, h) \left[\sum_{d:h\in d} w(d)\ell_n(d)\right] \\
&= \frac{1}{Z(f_i)} \sum_h c(\phi_k, h)\hat{\mathrm{D}}^n_\theta(h|f_i)
\end{aligned}$$

The quantity $\hat{\mathrm{D}}^n_\theta(h|f_i) = \sum_{d:h\in d} w(d)\ell_n(d)$ is the sum of the weight-length products of all derivations d containing hyperedge h. In the same way that $P_\theta(h|f_i)$ can be efficiently computed from inside and outside probabilities, this quantity $\hat{\mathrm{D}}^n_\theta(h|f_i)$ can be efficiently computed with two new inside and outside quantities, which we call $\hat{\mathrm{I}}_n(u)$ and $\hat{\mathrm{O}}_n(u)$. We provide recursions for these quantities in Figure 4. Like the standard inside and outside computations, these recursions run in time linear in the number of hyperedges in the forest.

While a full exposition of the algorithm is not possible in the available space, we give some brief intuition behind this dynamic program. We first define $\hat{\mathrm{I}}_n(u)$:

$$\hat{\mathrm{I}}_n(u) = \sum_{d_u} w(d_u)\ell_n(d)$$

where d_u is a derivation rooted at node u. This is a sum of weight-length products similar to $\hat{\mathrm{D}}$. To give a recurrence for $\hat{\mathrm{I}}$, we rewrite it:

$$\hat{\mathrm{I}}_n(u) = \sum_{d_u} \sum_{h\in d_u} [w(d_u)\ell_n(h)]$$

Here, we have broken up the total value of $\ell_n(d)$ across hyperedges in d. The bracketed quantity is a score of a *marked derivation* pair (d, h) where the edge h is some specific element of d. The score of a marked derivation includes the weight of the derivation and the factor $\ell_n(h)$ for the marked hyperedge.

This sum over marked derivations gives the inside recurrence in Figure 4 by the following decomposition. For $\hat{\mathrm{I}}_n(u)$ to sum over all marked derivation pairs rooted at u, we must consider two cases. First, the marked hyperedge could be at the root, in which case we must choose child derivations from regular inside scores and multiply in the local ℓ_n, giving the first summand of $\hat{\mathrm{I}}_n(u)$. Alternatively, the marked hyperedge is in exactly one of the children; for each possibility we recursively choose a marked derivation for one child, while the other children choose regular derivations. The second summand of $\hat{\mathrm{I}}_n(u)$ compactly expresses a sum over instances of this case. $\hat{\mathrm{O}}_n(u)$ decomposes similarly: the marked hyperedge could be local (first summand), under a sibling (second summand), or higher in the tree (third summand).

Once we have these new inside-outside quantities, we can compute $\hat{\mathrm{D}}$ as in Figure 5. This combination states that marked derivations containing h are either marked at h, below h, or above h.

As a final detail, computing the gradient $\nabla C^{\text{clip}}_n(\theta)$ (see the Appendix) involves a clipped version of the expected product of counts, for which a clipped $\hat{D}$ is required. This quantity can be computed with the same dynamic program with a slight modification. In Figure 4, we show the difference as a choice point when computing $\ell_n(h)$.

3.5 Implementation Details

As stated, the runtime of computing the required expectations for the objective and gradient is linear in the number of hyperedges in the forest. The

$$\hat{\mathrm{I}}_n(u) = \sum_{h \in \mathrm{IN}(u)} w(h) \left[\ell_n(h) \prod_{v \in tail(h)} \mathrm{I}(v) + \sum_{v \in tail(h)} \hat{\mathrm{I}}_n(v) \prod_{w \neq v} \mathrm{I}(w) \right]$$

$$\hat{\mathrm{O}}_n(u) = \sum_{h \in \mathrm{OUT}(u)} w(h) \left[\ell_n(h)\, \mathrm{O}(head(h)) \prod_{\substack{v \in tail(h) \\ v \neq u}} \mathrm{I}(v) \;+\; \mathrm{O}(head(h)) \sum_{\substack{v \in tail(h) \\ v \neq u}} \hat{\mathrm{I}}_n(v) \prod_{\substack{w \in tail(h) \\ w \neq v \\ w \neq u}} \mathrm{I}(w) \;+\; \hat{\mathrm{O}}_n(head(h)) \prod_{\substack{w \in tail(h) \\ w \neq u}} \mathrm{I}(w) \right]$$

$$\ell_n(h) = \begin{cases} \sum_{g_n} c(g_n, h) & \text{computing unclipped counts} \\ \sum_{g_n} c(g_n, h)\, \mathbb{1}\left[\mathbb{E}_\theta[c(g_n, d)] \leq c(g_n, r_i)\right] & \text{computing clipped counts} \end{cases}$$

Figure 4: Inside and Outside recursions for $\hat{\mathrm{I}}_n(u)$ and $\hat{\mathrm{O}}_n(u)$. $\mathrm{IN}(u)$ and $\mathrm{OUT}(u)$ refer to the incoming and outgoing hyperedges of u, respectively. $\mathrm{I}(\cdot)$ and $\mathrm{O}(\cdot)$ refer to standard inside and outside quantities, defined in Appendix Figure 7. We initialize with $\hat{\mathrm{I}}_n(u) = 0$ for all terminal forest nodes u and $\hat{\mathrm{O}}_n(root) = 0$ for the root node. $\ell_n(h)$ computes the sum of all n-grams of order n on a hyperedge h.

$$\hat{\mathrm{D}}^n_\theta(h|f_i) =$$

$$w(h) \left[\ell_n(h) \mathrm{O}(head(h)) \prod_{v \in tail(h)} \mathrm{I}(v) + \mathrm{O}(head(h)) \sum_{v \in tail(h)} \hat{\mathrm{I}}_n(v) \prod_{\substack{v \in tail(h) \\ w \neq v}} \mathrm{I}(w) + \hat{\mathrm{O}}_n(head(h)) \prod_{w \in tail(h)} \mathrm{I}(w) \right]$$

Figure 5: Calculation of $\hat{\mathrm{D}}^n_\theta(h|f_i)$ after $\hat{\mathrm{I}}_n(u)$ and $\hat{\mathrm{O}}_n(u)$ have been computed.

number of hyperedges is very large, however, because we must track n-gram contexts in the nodes, just as we would in an integrated language model decoder. These contexts are required both to correctly compute the model score of derivations and to compute clipped n-gram counts. To speed our computations, we use the cube pruning method of Huang and Chiang (2007) with a fixed beam size.

For regularization, we added an L_2 penalty on the size of θ to the CoBLEU objective, a simple addition for gradient ascent. We did not find that our performance varied very much for moderate levels of regularization.

3.6 Related Work

The calculation of expected counts can be formulated using the expectation semiring framework of Eisner (2002), though that work does not show how to compute expected products of counts which are needed for our gradient calculations. Concurrently with this work, Li and Eisner (2009) have generalized Eisner (2002) to compute expected products of counts on translation forests. The training algorithm of Kakade et al. (2002) makes use of a dynamic program similar to ours, though specialized to the case of sequence models.

4 Consensus Decoding

Once model parameters θ are learned, we must select an appropriate decoding objective. Several new decoding approaches have been proposed recently that leverage some notion of consensus over the many weighted derivations in a translation forest. In this paper, we adopt the fast consensus decoding procedure of DeNero et al. (2009), which directly complements CoBLEU tuning. For a source sentence f, we first build a translation forest, then compute the expected count of each n-gram in the translation of f under the model. We extract a k-best list from the forest, then select the translation that yields the highest BLEU score relative to the forest's expected n-gram counts. Specifically, let $\mathrm{BLEU}(e; r)$ compute the similarity of a sentence e to a reference r based on the n-gram counts of each. When training with CoBLEU, we replace e with expected counts and maximize θ. In consensus decoding, we replace r with expected counts and maximize e.

Several other efficient consensus decoding pro-

cedures would similarly benefit from a tuning procedure that aggregates over derivations. For instance, Blunsom and Osborne (2008) select the translation sentence with highest posterior probability under the model, summing over derivations. Li et al. (2009) propose a variational approximation maximizing sentence probability that decomposes over n-grams. Tromble et al. (2008) minimize risk under a loss function based on the linear Taylor approximation to BLEU, which decomposes over n-gram posterior probabilities.

5 Experiments

We compared CoBLEU training with an implementation of minimum error rate training on two language pairs.

5.1 Model

Our optimization procedure is in principle tractable for any syntactic translation system. For simplicity, we evaluate the objective using an Inversion Transduction Grammar (ITG) (Wu, 1997) that emits phrases as terminal productions, as in (Cherry and Lin, 2007). Phrasal ITG models have been shown to perform comparably to the state-of-the art phrase-based system Moses (Koehn et al., 2007) when using the same phrase table (Petrov et al., 2008).

We extract a phrase table using the Moses pipeline, based on Model 4 word alignments generated from GIZA++ (Och and Ney, 2003). Our final ITG grammar includes the five standard Moses features, an n-gram language model, a length feature that counts the number of target words, a feature that counts the number of monotonic ITG rewrites, and a feature that counts the number of inverted ITG rewrites.

5.2 Data

We extracted phrase tables from the Spanish-English and French-English sections of the Europarl corpus, which include approximately 8.5 million words of bitext for each of the language pairs (Koehn, 2002). We used a trigram language model trained on the entire corpus of English parliamentary proceedings provided with the Europarl distribution and generated according to the ACL 2008 SMT shared task specifications.[6] For tuning, we used all sentences from the 2007 SMT shared task up to length 25 (880 sentences for Spanish and 923 for French), and we tested on the subset of the first 1000 development set sentences which had length at most 25 words (447 sentences for Spanish and 512 for French).

[6] See `http://www.statmt.org/wmt08` for details.

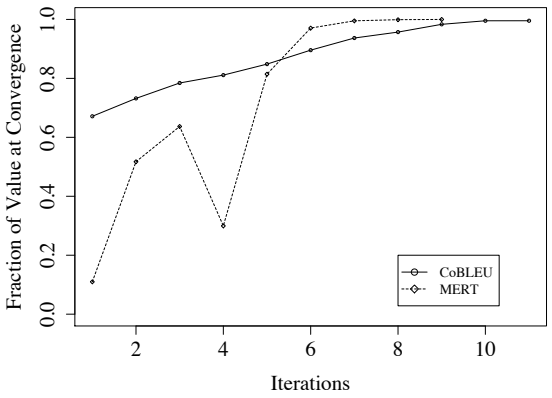


Figure 6: Trajectories of MERT and CoBLEU during optimization show that MERT is initially unstable, while CoBLEU training follows a smooth path to convergence. Because these two training procedures optimize different functions, we have normalized each trajectory by the final objective value at convergence. Therefore, the absolute values of this plot do *not* reflect the performance of either objective, but rather the smoothness with which the final objective is approached. The rates of convergence shown in this plot are not directly comparable. Each iteration for MERT above includes 10 iterations of coordinate ascent, followed by a decoding pass through the training set. Each iteration of CoBLEU training involves only one gradient step.

5.3 Tuning Optimization

We compared two techniques for tuning the nine log-linear model parameters of our ITG grammar. We maximized CoBLEU using gradient ascent, as described above. As a baseline, we maximized BLEU of the Viterbi translation derivations using minimum error rate training. To improve optimization stability, MERT used a cumulative k-best list that included all translations generated during the tuning process.

One of the benefits of CoBLEU training is that we compute expectations efficiently over an entire forest of translations. This has substantial stability benefits over methods based on k-best lists. In Figure 6, we show the progress of CoBLEU as compared to MERT. Both models are initialized from 0 and use the same features. This plot exhibits a known issue with MERT training: because new k-best lists are generated at each iteration, the objective function can change drastically between iterations. In contrast, CoBLEU converges

Consensus Decoding

	Spanish			
	Tune	Test	Δ	Br.
MERT	32.5	30.2	-2.3	0.992
CoBLEU	31.4	30.4	-1.0	0.992
MERT→CoBLEU	31.7	**30.8**	-0.9	0.992
	French			
	Tune	Test	Δ	Br.
MERT	32.5	31.1*	-1.4	0.972
CoBLEU	31.9	30.9	-1.0	0.954
MERT→CoBLEU	32.4	**31.2***	-0.8	0.953

Table 1: Performance measured by BLEU using a consensus decoding method over translation forests shows an improvement over MERT when using CoBLEU training. The first two conditions were initialized by 0 vectors. The third condition was initialized by the final parameters of MERT training. Br. indicates the brevity penalty on the test set. The * indicates differences which are not statistically significant.

smoothly to its final objective because the forests do not change substantially between iterations, despite the pruning needed to track n-grams. Similar stability benefits have been observed for lattice-based MERT (Macherey et al., 2008).

5.4 Results

We performed experiments from both French and Spanish into English under three conditions. In the first two, we initialized both MERT and CoBLEU training uniformly with zero weights and trained until convergence. In the third condition, we initialized CoBLEU with the final parameters from MERT training, denoted MERT→CoBLEU in the results tables. We evaluated each of these conditions on both the tuning and test sets using the consensus decoding method of DeNero et al. (2009). The results appear in Table 1.

In Spanish-English, CoBLEU slightly outperformed MERT under the same initialization, while the opposite pattern appears for French-English. The best test set performance in both language pairs was the third condition, in which CoBLEU training was initialized with MERT. This condition also gave the highest CoBLEU objective value. This pattern indicates that CoBLEU is a useful objective for translation with consensus decoding, but that the gradient ascent optimization is getting stuck in local maxima during tuning. This issue can likely be addressed with annealing, as described in (Smith and Eisner, 2006).

Interestingly, the brevity penatly results in French indicate that, even though CoBLEU did not outperform MERT in a statistically significant way, CoBLEU tends to find shorter sentences with higher n-gram precision than MERT.

Viterbi Decoding

	Spanish		
	Tune	Test	Δ
MERT	32.5	30.2	-2.3
MERT→CoBLEU	30.5	30.9	**+0.4**
	French		
	Tune	Test	Δ
MERT	32.0	31.0	-1.0
MERT→CoBLEU	31.7	30.9	-0.8

Table 2: Performance measured by BLEU using *Viterbi* decoding indicates that CoBLEU is less prone to overfitting than MERT.

Table 1 displays a second benefit of CoBLEU training: compared to MERT training, CoBLEU performance degrades less from tuning to test set. In Spanish, initializing with MERT-trained weights and then training with CoBLEU actually decreases BLEU on the tuning set by 0.8 points. However, this drop in tuning performance comes with a corresponding increase of 0.6 on the test set, relative to MERT training. We see the same pattern in French, albeit to a smaller degree.

While CoBLEU ought to outperform MERT using consensus decoding, we expected that MERT would give better performance under Viterbi decoding. Surprisingly, we found that CoBLEU training actually outperformed MERT in Spanish-English and performed equally well in French-English. Table 2 shows the results. In these experiments, we again see that CoBLEU overfit the training set to a lesser degree than MERT, as evidenced by a smaller drop in performance from tuning to test set. In fact, test set performance actually improved for Spanish-English CoBLEU training while dropping by 2.3 BLEU for MERT.

6 Conclusion

CoBLEU takes a fundamental quantity used in consensus decoding, expected n-grams, and trains to optimize a function of those expectations. While CoBLEU can therefore be expected to increase test set BLEU under consensus decoding, it is more surprising that it seems to better regularize learning even for the Viterbi decoding condition. It is also worth emphasizing that the CoBLEU approach is applicable to functions of expected n-gram counts other than BLEU.

Appendix: The Gradient of CoBLEU

We would like to compute the gradient of

$$\left(1 - \frac{|R|}{\sum_{i=1}^{m}\sum_{g_1}\mathbb{E}_\theta[c(g_1,d)|f_i]}\right)_-$$

$$+\frac{1}{4}\sum_{n=1}^{4}\ln\frac{\sum_{i=1}^{m}\sum_{g_n}\min\{\mathbb{E}_\theta[c(g_n,d)|f_i], c(g_n,r_i)\}}{\sum_{i=1}^{m}\sum_{g_n}\mathbb{E}_\theta[c(g_n,d)|f_i]}$$

To simplify notation, we introduce the functions

$$C_n(\theta) = \sum_{i=1}^{m}\sum_{g_n}\mathbb{E}_\theta[c(g_n,e)|f_i]$$

$$C_n^{\text{clip}}(\theta) = \sum_{i=1}^{m}\sum_{g_n}\min\{\mathbb{E}_\theta[c(g_n,d)|f_i], c(r,g_n)\}$$

$C_n(\theta)$ represents the sum of the expected counts of all n-grams or order n in all translations of the source corpus F, while $C_n^{\text{clip}}(\theta)$ represents the sum of the same expected counts, but clipped with reference counts $c(g_n, r_i)$.

With this notation, we can write our objective function CoBLEU(R, F, θ) in three terms:

$$\left(1 - \frac{|R|}{C_1(\theta)}\right)_-$$

$$+\frac{1}{4}\sum_{n=1}^{4}\ln C_n^{\text{clip}}(\theta) - \frac{1}{4}\sum_{n=1}^{4}\ln C_n(\theta)$$

We first state an identity:

$$\sum_{g_n}\frac{\partial}{\partial\theta_k}\mathbb{E}_\theta[c(g_n,d)|f_i] =$$

$$\mathbb{E}_\theta\left[c(\phi_k,d)\cdot\ell_n(d)|f_i\right]$$
$$-\mathbb{E}_\theta\left[\ell_n(d)|f_i\right]\cdot\mathbb{E}_\theta[c(\phi_k,d)|f_i]$$

which can be derived by expanding the expectation on the left-hand side

$$\sum_{g_n}\sum_{d}\frac{\partial}{\partial\theta_k}P_\theta(d|f_i)c(g_n,d)$$

and substituting

$$\frac{\partial}{\partial\theta_k}P_\theta(d|f_i) =$$

$$P_\theta(d|f_i)c(\phi_k,d) - P_\theta(d|f_i)\sum_{d'}P_\theta(d'|f_i)c(\phi_k,d')$$

Using this identity and some basic calculus, the gradient $\nabla C_n(\theta)$ is

$$\sum_{i=1}^{m}\mathbb{E}_\theta\left[c(\phi_k,d)\cdot\ell_n(d)|f_i\right] - C_n(\theta)\mathbb{E}_\theta[c(\phi_k,d)|f_i]$$

$$\begin{aligned}
\mathrm{I}(u) &= \sum_{h\in\mathrm{IN}(u)} w(h)\left[\prod_{v\in tail(h)}\mathrm{I}(v)\right] \\
\mathrm{O}(u) &= \sum_{h\in OUT(u)} w(h)\left[\mathrm{O}(head(h))\prod_{\substack{v\in tail(h)\\ v\neq u}}\mathrm{I}(v)\right]
\end{aligned}$$

Figure 7: Standard Inside-Outside recursions which compute $\mathrm{I}(u)$ and $\mathrm{O}(u)$. $\mathrm{IN}(u)$ and $\mathrm{OUT}(u)$ refer to the incoming and outgoing hyperedges of u, respectively. We initialize with $\mathrm{I}(u) = 1$ for all terminal forest nodes u and $\mathrm{O}(root) = 1$ for the root node. These quantities are referenced in Figure 4.

and the gradient $\nabla C_n^{\text{clip}}(\theta)$ is given by

$$\sum_{i=1}^{m}\sum_{g_n}\Bigg[\mathbb{E}_\theta\left[c(g_n,d)\cdot c(\phi_k,d)|f_i\right]$$

$$\cdot\mathbb{1}\Big[\mathbb{E}_\theta[c(g_n,d)|f_i]\leq c(g_n,r_i)\Big]\Bigg]$$

$$-C_n^{\text{clip}}(\theta)\mathbb{E}_\theta[c(\phi_k,d)+f_i]$$

where $\mathbb{1}$ denotes an indicator function. At the top level, the gradient of the first term (the brevity penalty) is

$$\frac{|R|\nabla C_1(\theta)}{C_1(\theta)^2}\mathbb{1}\Big[C_1(\theta)\leq|R|\Big]$$

The gradient of the second term is

$$\frac{1}{4}\sum_{n=1}^{4}\frac{\nabla C_n^{\text{clip}}(\theta)}{C_n^{\text{clip}}(\theta)}$$

and the gradient of the third term is

$$-\frac{1}{4}\sum_{n=1}^{4}\frac{\nabla C_n(\theta)}{C_n(\theta)}$$

Note that, because of the indicator functions, CoBLEU is non-differentiable when $\mathbb{E}_\theta[c(g_n,d)|f_i] = c(g_n,r_i)$ or $C_n(\theta) = |R|$. Formally, we must compute a sub-gradient at these points. In practice, we can choose between the gradients calculated assuming the indicator function is 0 or 1; we always choose the latter.

References

Phil Blunsom and Miles Osborne. 2008. Probabilistic inference for machine translation. In *Proceedings of the Conference on Emprical Methods for Natural Language Processing*.

Colin Cherry and Dekang Lin. 2007. Inversion transduction grammar for joint phrasal translation modeling. In *The Annual Conference of the North American Chapter of the Association for Computational Linguistics Workshop on Syntax and Structure in Statistical Translation*.

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

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*.

John DeNero, David Chiang, and Kevin Knight. 2009. Fast consensus decoding over translation forests. In *The Annual Conference of the Association for Computational Linguistics*.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *The Annual Conference of the Association for Computational Linguistics*.

Sham Kakade, Yee Whye Teh, and Sam T. Roweis. 2002. An alternate objective function for markovian fields. In *Proceedings of ICML*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *The Annual Conference of the Association for Computational Linguistics*.

Philipp Koehn. 2002. Europarl: A multilingual corpus for evaluation of machine translation.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum Bayes-risk decoding for translation hypergraphs and lattices. In *The Annual Conference of the Association for Computational Linguistics*.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009. Variational decoding for statistical machine translation. In *The Annual Conference of the Association for Computational Linguistics*.

W. Macherey, F. Och, I. Thayer, and J. Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *In Proceedings of Empirical Methods in Natural Language Processing*.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL)*, pages 160–167, Morristown, NJ, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *The Annual Conference of the Association for Computational Linguistics*.

Slav Petrov, Aria Haghighi, and Dan Klein. 2008. Coarse-to-fine syntactic machine translation using language projections. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 108–116, Honolulu, Hawaii, October. Association for Computational Linguistics.

David Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *In Proceedings of the Association for Computational Linguistics*.

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice minimum Bayes-risk decoding for statistical machine translation. In *The Conference on Empirical Methods in Natural Language Processing*.

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

Using Word-Sense Disambiguation Methods to Classify Web Queries by Intent

Emily Pitler
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
epitler@seas.upenn.edu

Ken Church
Johns Hopkins University
Human Language Technology Center of Excellence
Baltimore, MD 21211
Kenneth.Church@jhu.edu

Abstract

Three methods are proposed to classify queries by intent (CQI), e.g., navigational, informational, commercial, etc. Following mixed-initiative dialog systems, search engines should distinguish navigational queries where the user is taking the initiative from other queries where there are more opportunities for system initiatives (e.g., suggestions, ads). The query intent problem has a number of useful applications for search engines, affecting how many (if any) advertisements to display, which results to return, and how to arrange the results page. Click logs are used as a substitute for annotation. Clicks on ads are evidence for commercial intent; other types of clicks are evidence for other intents. We start with a simple Naïve Bayes baseline that works well when there is plenty of training data. When training data is less plentiful, we back off to nearby URLs in a click graph, using a method similar to Word-Sense Disambiguation. Thus, we can infer that *designer trench* is commercial because it is close to *www.saksfifthavenue.com*, which is known to be commercial. The baseline method was designed for precision and the backoff method was designed for recall. Both methods are fast and do not require crawling webpages. We recommend a third method, a hybrid of the two, that does no harm when there is plenty of training data, and generalizes better when there isn't, as a strong baseline for the CQI task.

1 Classify Queries By Intent (CQI)

Determining query intent is an important problem for today's search engines. Queries are short (consisting of 2.2 terms on average (Beitzel et al., 2004)) and contain ambiguous terms. Search engines need to derive what users want from this limited source of information. Users may be searching for a specific page, browsing for information, or trying to buy something. Guessing the correct intent is important for returning relevant items. Someone searching for *designer trench* is likely to be interested in results or ads for trench coats, while someone searching for *world war I trench* might be irritated by irrelevant clothing advertisements.

Broder (2002) and Rose and Levinson (2004) categorized queries into those with *navigational*, *informational*, and *transactional* or *resource-seeking* intent. Navigational queries are queries for which a user has a particular web page in mind that they are trying to navigate to, such as *greyhound bus*. Informational queries are those like *San Francisco*, in which the user is trying to gather information about a topic. Transactional queries are those like *digital camera* or *download adobe reader*, where the user is seeking to make a transaction or access an online resource.

Knowing the intent of a query greatly affects the type of results that are relevant. For many queries, Wikipedia articles are returned on the first page of results. For informational queries, this is usually appropriate, as a Wikipedia article contains summaries of topics and links to explore further. However, for navigational or transactional queries, Wikipedia is not as appropriate. A user looking for the *greyhound bus* homepage is probably not interested in facts about the company. Similarly, someone looking to *download adobe reader* will not be interested in Wikipedia's description of the product's history. Conversely, for informational queries, Wikipedia articles tend to be appropriate while advertisements are not. The user searching for *world war I trench* might find the Wikipedia article on trench warfare useful, while he is prob-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1428–1436,
Singapore, 6-7 August 2009.

(a) The advertisements and related searches are probably more likely to be clicked on than the top result for *designer trench*.

(b) The top result will receive more clicks than the spelling suggestion. Wikipedia often receives lots of clicks, but not for commercial queries like *bestbuy*.

Figure 1: Results pages from two major search engines. A search results page has limited real estate that must be divided between search results, spelling suggestions, query suggestions, and ads.

ably not interested in purchasing clothing, or even World War I related products. We noticed empirically that queries in the logs tend to have a high proportion of clicks on the Wikipedia article or the ads, but almost never both. The Wikipedia page for Best Buy in Figure 1(b) is probably a waste of space. Knowing whether a particular query is navigational, informational, or transactional would improve search and advertising relevance.

After a query is issued, search engines return a list of results, and possibly also advertisements, suggestions of related searches, and spelling suggestions. For different queries, these alternatives have varying utilities to the users. Consider the queries in Figures 1(a) and 1(b). For *designer trench*, the advertisements may well be more useful to the user than the standard set of results. The query suggestions for *designer trench* all would help refine the query, whereas the suggestions for *bestbuy* are less useful, as they would either return the same set of results or take the user to Best Buy's competitors' sites. The spelling suggestion for *best buy* instead of *bestbuy* is also unnecessary. Devoting more page space to the content that is likely to be clicked on could help improve the user experience.

In this paper we consider the task of: given a class of queries, which types of answer (standard search, ads, query suggestions, or spelling sug-

gestions) are likely to be clicked on? Typos will tend to have more clicks on the spelling suggestions, informational queries will have more clicks on Wikipedia pages, and commercial queries will have more clicks on the ads. The observed behavior of where users click tells us something about the hidden intentions of the users when they issue that query.

We focus on *commercial intent* (Dai et al., 2006), the intent to purchase a product or service, to illustrate our method of predicting query intent. The business model of web search today is heavily dependent on advertising. Advertisers bid on queries, and then the search results page also contains "sponsored" sites by the advertisers who won the auction for that query. It is thus advantageous for the advertisers to bid on queries which are most likely to result in a commercial transaction. If a query is classified as likely implying commercial intent, but the advertisers have overlooked this query, then the search engine may want to suggest that advertisers bid on that query. The search engine may also want to treat queries classified as having commercial intent differently, by rearranging the appearance of the page, or by showing more or fewer advertisements.

This paper starts with a simple Naïve Bayes baseline to classify queries by intent (CQI). Supervised methods work well, especially when there is plenty of annotated data for testing and training. Unfortunately, since we don't have as much annotated data as we might like, we propose two workarounds:

1. Use click logs as a substitute for annotated data. Clicks on ads are evidence for commercial intent; other types of clicks are evidence for other intents.

2. We propose a method similar to Yarowsky (1995) to generalize beyond the training set.

2 Related Work

Click logs have been used for a variety of tasks involved in information retrieval, including predicting which pages are the best results for queries (Piwowarski and Zaragoza, 2007; Joachims, 2002; Xue et al., 2004), choosing relevant advertisements (Chakrabarti et al., 2008), suggesting related queries (Beeferman and Berger, 2000), and personalizing results (Tan et al., 2006). Queries that have a navigational intent tended to have a highly skewed click distribution, while users clicked on a wider range of results after issuing informational queries. Lee et al. (2005) used the click distributions to classify navigational versus informational intents.

While navigational, informational, and resource-seeking are very broad intentions, other researchers have looked at personalization and intent on a per user basis. Downey et al. (2008) use the last URL visited in a session or the last search engine result visited as a proxy for the user's information goal, and then looked at the correspondence between information needs and queries (how the goals are expressed).

We are interested in a granularity of intent in between navigational/informational/resource-seeking and personalized intents. For these sorts of intents, the web pages associated with queries provide useful information. To classify queries into an ontology of commercial queries, Broder et al. (2007) found that a classifier that used the text of the top result pages performed much better than a classifier that used only the query string. While the results are quite good on their hierarchy of 6000 types of commercial intents, they manually constructed about 150 hand-picked examples each for each of the 6000 intents. Beitzel et al. (2005) do semi-supervised learning over the query logs to classify queries into topics, but also train with hundreds of thousands of manually annotated queries. Thus, while we also use the query logs and the identities of web pages of associated with each query, we are interested in finding methods that can be applied when that much annotation is prohibitive.

Semi-supervised methods over the click graph make it possible to train classifiers after starting from a much smaller set of seed queries. Li et al. (2008) used the semi-supervised learning method described in Zhou et al. (2004) to gain a much larger training set of examples, and then trained classifiers for product search or job search on the expanded set. Random walk methods over the click graph have also been used to propagate relations between URLs, for tasks such as finding "adult" content (Craswell and Szummer, 2007) and suggesting related queries (Antonellis et al., 2008) and content (Baluja et al., 2008). In our work we also seek to classify query intent using the click graph, but we demonstrate the effectiveness of a simple method by building deci-

sion lists of URLs. In addition, we evaluate our method automatically by using user click rates, rather than assembling hand-labeled examples for training and testing.

Dai et al. (2006) also classified queries by commercial intent, but their method involved crawling the top landing pages for each query, which can be quite time-consuming. In this paper we investigate the commercial intent problem when crawling pages is not feasible, and use only the identities of the top URLs.

3 Using Click Logs as a Substitute for Annotation

Prior work has used click logs in lieu of manual annotations of relevance ratings, either of webpages (Joachims, 2002) or of sponsored search advertisements (Ciaramita et al., 2008). Here we use the click logs as a large-scale source of intents. Logs from Microsoft's Live Search are used for training and test purposes. Logs from May 2008 were used for training, and logs from June 2008 were used for testing.

The logs distinguish four types of clicks: (a) search results, (b) ads, (c) spelling suggestions and (d) query suggestions. Some prototypical queries of each type are shown in Table 1. As mentioned above, clicks on ads are evidence for commercial intent; other types of clicks are evidence for other intents. The query, *ebay official*, is assumed to be commercial intent, because a large fraction of the clicks are on ads. In contrast, typos tend to have relatively more clicks on "did-you-mean" spelling suggestions.

The query logs contain over a terabyte of data for each day, and our experiments were done using months of logs at a time. We used SCOPE (Chaiken et al., 2008), a scripting programming language designed for doing Map-Reduce (Dean and Ghemawat, 2004) style computations, to distribute the task of aggregating the counts of each query over thousands of servers. As the same query is often issued several times by multiple users across an entire month of search logs, we summarize each query with four ratios–search results clicks:overall clicks, ad clicks:overall clicks, spelling suggestion clicks:overall clicks, and query suggestion clicks:overall clicks.

A couple of steps were taken to ensure reliable ratios. We are classifying types, not tokens, and so limited ourselves to those queries with 100 or more clicks. This still leaves us with over half a million distinct queries for training and for testing, yet allows us to use click ratios as a substitute for annotating these huge data sets. If a query was only issued once and the user clicked on an ad, that may be more a reflection of the user, rather than reflecting that the query is 100% commercial. In addition, the ratios compare clicks of one type with clicks of another, rather than comparing clicks with impressions. There is less risk of a failure to find fallacy if we count events (clicks) instead of non-events (non-clicks). There are many reasons for non-clicks, only some of which tell us about the meaning of the query. There are bots that crawl pages and never click. Many links can't be seen (e.g., if they are below the fold).

Queries are labeled as positive examples of commercial intent if their ratio is in the top half of the training set, and negative otherwise. A similar procedure is used to label queries with the three other intents.

Our task is to predict future click patterns based on past click patterns. Note that a query may appear in both the test set and the training set, although not necessarily with the same label. In fact, because of the robustness requirement of 100+ clicks, many queries appear in both sets; 506,369 out of 591,122 of the test queries were also present in the training month. The overlap reflects natural processes on the web, with a long tail (of queries that will never be seen again) and a big fat head (of queries that come up again and again). Throwing away the overlap would both drastically reduce the size of the data and make the problem less realistic for a commercial application.

We therefore report results on various training set sizes so that we can show both: (a) the ability of the proposed method to generalize to unseen queries, and (b) the high performance of the baselines in a realistic setting. We vary the number of new queries by training the methods on subsets of 20%, 40%, 60%, 80%, and 100% of the positive examples (along with all the negative examples) in the training set. This led to the test set having 17%, 34%, 52%, 67%, and 86% actual overlap of these queries, respectively, with the training sets.

Click Type (Area on Results Page)	Query Type (Intent)	Example
Spelling Suggestion	Typo	www.lastmintue.com.au
Ad	Commercial Intent	ebay official
Query Suggestion	Suggestible	sears employees (where there are some popular query suggestions indicating how current employees can navigate to the benefits site, as well as how others can apply for employment)
Search Result	Standard Search	craigslist, denver, co

Table 1: Queries with a high percentage of clicks in each category

4 Three CQI Methods

4.1 Method 1: Look-up Baseline

The baseline method checks if a query was present in the training set, and if so, outputs the label from the training set. If the query was not present, it backs off to the appropriate default label: "non-commercial" for the commercial intent task (and "non-suggestible", "not a typo", etc. for the other CQI tasks). This very simple baseline method is effective because the ratios tend to be fairly stable from one month to the next. The query, *ebay official*, for example, has relatively high ad clicks in both the training month as well as the test month. The next section will propose an alternative method to address the main weakness of the baseline method, the inability to generalize beyond the queries in the training set.

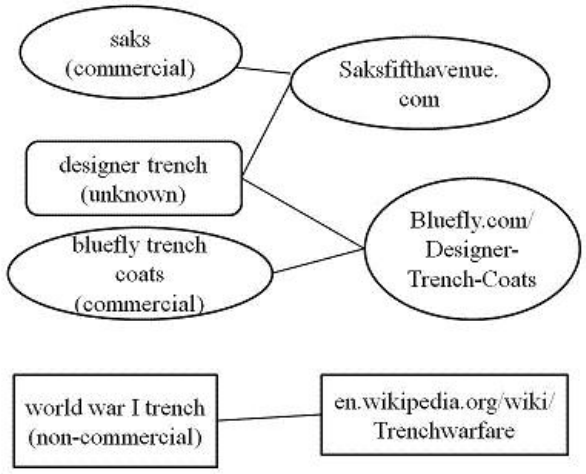


Figure 2: *saks* and *bluefly trench coats* are known to be commercial, while *world war I trench* is known to be non-commercial. What about *designer trench*? We can classify it as commercial because it shares URLs with the known commercial queries.

4.2 Method 2: Using Click Graph Context to Generalize Beyond the Queries in the Training Set

To address the generalization concern, we propose a method inspired by Yarowsky (1994). Word sense disambiguation is a classic problem in natural language processing. Some words have multiple senses; for instance, *bank* can either mean a riverbank or a financial institution, and for various tasks such as information retrieval, parsing, or information extraction, it is useful to be able to differentiate between the possible meanings.

When a word is being used in each sense, it tends to appear in a different context. For example, if the word *muddy* is nearby *bank*, the author is probably using the riverbank sense of the term, while if the word *deposit* is nearby, the word is probably being used with the financial sense.

Yarowksy (1995) thus creates a list of each possible context, sorted by how strong the evidence is for a particular sense. To classify a new example, Yarowsky (1994) finds the most informative collocation pattern that applies to the test example.

In this work, rather than using the surrounding words as context as in text classification, we consider the surrounding URLs in the click graph as context. A sample portion of the click graph is shown in figure 2. The figure shows queries on the left and URLs on the right. The click graph was computed on a very large sample of logs computed well before the training period. There is an edge from a query q to a URL u if at least 10 users issued q and then clicked on u.

For each URL, we look at its neighboring queries and calculate the log likelihood ratio of their labels in the training set. We classify a new query q according to URL^*, the neighboring URL with the strongest opinion (highest absolute value of the log likelihood ratio). That is, we compute URL^* with:

$$\text{argmax}_{U_i \in Nbr(q)} \left| \log \frac{\Pr(Intent|U_i)}{\Pr(\neg Intent|U_i)} \right|$$

If the neighboring opinion is positive (that is, $\Pr(Intent|URL^*) > \Pr(\neg Intent|URL^*)$), then the query q is assigned a positive label. Otherwise, q is assigned a negative label.

In Figure 2, we classify *designer trench* as a commercial query based on the neighbor with the strongest opinion. In this case, there was a tie between two neighbors with equally strong opinions: *www.saksfifthavenue.com* and *www.bluefly.com/Designer-Trench-Coats*. Both neighbors are associated with queries that were labeled commercial in the training set: *saks* and *bluefly trench coats*, respectively.

This method allows the labels of training set queries to propagate through the URLs to new test set queries.

4.3 Method 3: Hybrid ("Better Together")

We recommend a hybrid of the two methods:

- Method 1: the look-up baseline
- Method 2: use click graph context to generalize beyond the queries in the training set

Method 1 is designed for precision and method 2 is designed for recall. The hybrid uses method 1 when applicable, and otherwise, backs off to method 2.

5 Results

5.1 Commercial Intent

Table 2 and Figures 3(a) and 3(b) compare the performance on the proposed hybrid method with the baseline. When there is plenty of training material, both methods perform about equally well (the look-up baseline has an F-score of 84.1%, compared with the hybrid method's F-score of 85.3%), but generalization becomes important when training data is severely limited. Figure 3(a) shows that the proposed method does no harm and might even help a little when there is plenty of training data. The hybrid's main benefit is generalization to queries beyond the training set. If we severely limit the size of the training set to just 20% of the month, as in Figure 3(b), then the proposed hybrid method is substantially better than the baseline. In this case, the proposed hybrid method's F-score is 65.8%, compared with the look-up method's F-score of 28.4%.

5.2 Other types of clicks

Table 3 and Figures 4(a) and 4(b) show a similar pattern for the query suggestion task. In fact, the pattern is perhaps even stronger for the query suggestion task than commercial intent. When the full training set is used, the hybrid method achieves an F-score of 91.9% (precision = 91.5%, recall = 92.3%). When only 20% of the training data is used, the hybrid method has an F-score of 73.9%, compared with the baseline's F-score of 29.6%. A similar pattern was observed for clicks on search results.

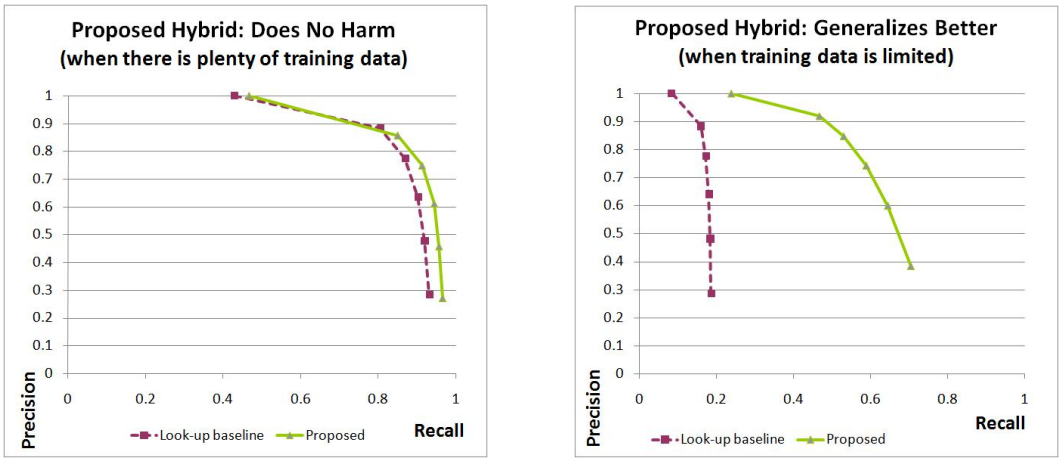
The one exception is the spelling suggestion task, where the context heuristic proved ineffective, for reasons that should not be surprising in retrospect. Click graph distance is an effective heuristic for many intents, but not for typos. Users who issue misspelled the query have the same goals as users who correctly spell the query, so we shouldn't expect URLs to be able to differentiate them. For misspelled queries, for example, *yuotube*, there are correctly spelled queries, like *youtube*, with the same intent that will tend to be associated with the same set of URLs (such as *www.youtube.com*).

6 Conclusion and Future Work

We would like to be able to distinguish web queries by intent. Unfortunately, we don't have annotated data for query intent, but we do have access to large quantities of click logs. The logs distinguish four types of clicks: (a) search results, (b) ads, (c) spelling suggestions and (d) query suggestions. Clicks on ads are evidence for commercial intent; other types of clicks are evidence for other intents. Click logs are huge sources of data, and while there are privacy concerns, anonymized logs are beginning to be released for research purposes (Craswell et al., 2009).

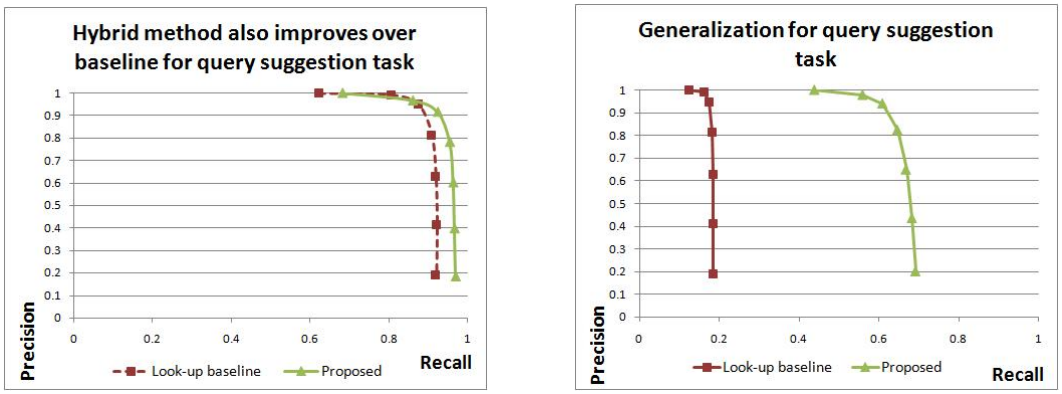
Besides commercial intent, queries can also be divided into two broader classes: queries in which the user is browsing and queries for which the user is navigating. Clicks on the ads and query suggestions indicate that users are browsing and willing to look at these alternative suggestions, while clicks on the search results indicate that the users were navigating to what they were searching for. Clicks on typos indicate neither, as presumably the users are not entering typos on purpose.

Just as dialogue management systems learn policies for when to allow *user* initiative (the user



(a) (b)

Figure 3: Better together: proposed hybrid is no worse than baseline (left) and generalizes better to unseen tail queries (right). The two panels are the same, except that the training set was reduced on the right to test generalization error.



(a) (b)

Figure 4: Similar to Figures 3(a) and 3(b), adding the decision list method generalizes over the look-up method for the "suggestible" task.

can respond in an open way) versus *system* initiative (the system asks the user questions with a restricted set of possible answers) (Relaño et al., 1999; Scheffler and Young, 2002; Singh et al., 2002), search engines may want to learn policies for when the user just wants the search results or when the user is open to suggestions. When users want help (they want the search engine to suggest results), more space on the page should be devoted to the ads and the query suggestions. When the users know what it is they want, more of the page should be given to the search results they asked for.

We started with a simple baseline for predicting click location that had great precision, but didn't generalize well beyond the queries in the training set. To improve recall, we proposed a context heuristic that backs off in the click graph. The backoff method is similar to Yarowsky's Word Sense Disambiguation method, except that context is defined in terms of URLs nearby in click graph distance, as opposed to words nearby in the text.

Our third method, a hybrid of the baseline method and the backoff method, is the strongest baseline we have come up with. The evaluation showed that the hybrid does no harm when there is plenty of training data, and generalizes better when there isn't.

A direction for further research would be to see if propagating query intent through URLs that are not direct neighbors but are further away, perhaps through random walk methods (Baluja et al., 2008;

Training Size	F-score			Precision / Recall		
	Baseline	Method 2	Hybrid	Baseline	Method 2	Hybrid
100%	84.1	75.6	**85.3**	88.2 / 80.4	76.6 / 74.6	85.7 / 85.0
80%	74.4	74.8	**83.5**	88.2 / 64.3	79.3 / 70.7	86.7 / 80.6
60%	62.4	72.9	**80.7**	88.3 / 48.2	82.5 / 65.3	87.9 / 74.6
40%	47.9	70.1	**76.0**	77.5 / 34.7	78.5 / 63.3	80.7 / 66.0
20%	28.4	62.5	**65.8**	77.6 / 17.4	75.9 / 53.1	74.3 / 59.1

Table 2: The baseline and hybrid methods have comparable F-scores when there is plenty of training data, but generalization becomes important when training data is severely limited. The proposed hybrid method generalizes better as indicated by the widening gap in F-scores with smaller and smaller training sets.

Training Size	F-score			Precision / Recall		
	Baseline	Method 2	Hybrid	Baseline	Method 2	Hybrid
100%	91.0	86.2	**91.9**	94.9 / 87.4	90.7 / 82.3	91.5 / 92.3
80%	80.5	85.2	**90.6**	94.9 / 69.9	91.6 / 79.7	91.9 / 89.4
60%	67.6	83.3	**88.6**	94.9 / 52.4	92.6 / 75.8	92.3 / 85.1
40%	51.0	79.5	**84.7**	94.9 / 34.9	87.6 / 72.7	93.0 / 77.8
20%	29.6	69.8	**73.9**	81.5 / 18.1	90.6 / 56.8	94.0 / 60.8

Table 3: F-scores on the query suggestion task. As in the commercial intent task, the proposed hybrid method does no harm when there is plenty of training data, but generalizes better when training data is severely limited.

Antonellis et al., 2008) improves classification.

Similar methods could be applied in future work to many other applications such labeling queries and URLs by: language, market, location, time, intended for a search vertical (such as medicine, recipes), intended for a type of answer (maps, pictures), as well as inappropriate intent (porn, spam).

In addition to click type, there are many other features in the logs that could prove useful for classifying queries by intent, e.g., who issued the query, when and where. Similar methods could also be used to personalize search (Teevan et al., 2008); for queries that mean different things to different people, the Yarowsky method could be applied to variables such as user, time and place, so the results reflect what a particular user intended in a particular context.

7 Acknowledgments

We thank Sue Dumais for her helpful comments on an early draft of this work. We would also like to thank the members of the Text Mining, Search, and Navigation (TMSN) group at Microsoft Research for useful discussions and the anonymous reviewers for their helpful comments.

References

I. Antonellis, H. Garcia-Molina, and C.C. Chang. 2008. Simrank++: query rewriting through link analysis of the clickgraph (poster). *WWW*.

S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. 2008. Video suggestion and discovery for youtube: taking random walks through the view graph. *WWW*.

D. Beeferman and A. Berger. 2000. Agglomerative clustering of a search engine query log. In *SIGKDD*, pages 407–416.

S.M. Beitzel, E.C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. 2004. Hourly analysis of a very large topically categorized web query log. *SIGIR*, pages 321–328.

S.M. Beitzel, E.C. Jensen, O. Frieder, D.D. Lewis, A. Chowdhury, and A. Kolcz. 2005. Improving automatic query classification via semi-supervised learning. *ICDM*, pages 42–49.

A.Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. 2007. Robust classification of rare queries using web knowledge. *SIGIR*, pages 231–238.

A. Broder. 2002. A taxonomy of web search. *SIGIR*, 36(2).

R. Chaiken, B. Jenkins, P.Å. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. 2008. SCOPE:

Easy and efficient parallel processing of massive data sets. *Proceedings of the VLDB Endowment archive*, 1(2):1265–1276.

D. Chakrabarti, D. Agarwal, and V. Josifovski. 2008. Contextual advertising by combining relevance with click feedback. *WWW*.

M. Ciaramita, V. Murdock, and V. Plachouras. 2008. Online learning from click data for sponsored search.

N. Craswell and M. Szummer. 2007. Random walks on the click graph. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 239–246.

N. Craswell, R. Jones, G. Dupret, and E. Viegas (Conference Chairs). 2009. Wscd '09: Proceedings of the 2009 workshop on web search click data.

H.K. Dai, L. Zhao, Z. Nie, J.R. Wen, L. Wang, and Y. Li. 2006. Detecting online commercial intention (OCI). *WWW*, pages 829–837.

J. Dean and S. Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. *OSDI*, pages 137–149.

D. Downey, S. Dumais, D. Liebling, and E. Horvitz. 2008. Understanding the relationship between searchers' queries and information goals. In *CIKM*.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM*.

U. Lee, Z. Liu, and J. Cho. 2005. Automatic identification of user goals in Web search. In *WWW*, pages 391–400.

X. Li, Y.Y. Wang, and A. Acero. 2008. Learning query intent from regularized click graphs. In *SIGIR*, pages 339–346.

B. Piwowarski and H. Zaragoza. 2007. Predictive user click models based on click-through history. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 175–182.

J. Relaño, D. Tapias, M. Rodríguez, M. Charfuelán, and L. Hernández. 1999. Robust and flexible mixed-initiative dialogue for telephone services. In *Proceedings of EACL*.

D.E. Rose and D. Levinson. 2004. Understanding user goals in web search. *WWW*, pages 13–19.

K. Scheffler and S. Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of HLT*, pages 12–19.

S. Singh, D. Litman, M. Kearns, and M. Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16(1):105–133.

Bin Tan, Xuehua Shen, and ChengXiang Zhai. 2006. Mining long-term search history to improve search accuracy. pages 718–723. KDD.

J. Teevan, S.T. Dumais, and D.J. Liebling. 2008. To personalize or not to personalize: modeling queries with variation in user intent. *SIGIR*, pages 163–170.

Gui-Rong Xue, Hua-Jun Zeng, Zheng Chen, Yong Yu, Wei-Ying Ma, WenSi Xi, and WeiGuo Fan. 2004. Optimizing web search using web clickthrough data. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 118–126.

D. Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. *ACL*, pages 88–95.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. *ACL*, pages 189–196.

D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Scholkopf. 2004. Learning with Local and Global Consistency. In *NIPS*.

Semi-Supervised Learning for Semantic Relation Classification using Stratified Sampling Strategy

Longhua Qian Guodong Zhou Fang Kong Qiaoming Zhu
Jiangsu Provincial Key Lab for Computer Information Processing Technology
School of Computer Science and Technology, Soochow University
1 Shizi Street, Suzhou, China 215006
{qianlonghua,gdzhou,kongfang,qmzhu}@suda.edu.cn

Abstract

This paper presents a new approach to selecting the initial seed set using stratified sampling strategy in bootstrapping-based semi-supervised learning for semantic relation classification. First, the training data is partitioned into several strata according to relation types/subtypes, then relation instances are randomly sampled from each stratum to form the initial seed set. We also investigate different augmentation strategies in iteratively adding reliable instances to the labeled set, and find that the bootstrapping procedure may stop at a reasonable point to significantly decrease the training time without degrading too much in performance. Experiments on the ACE RDC 2003 and 2004 corpora show the stratified sampling strategy contributes more than the bootstrapping procedure itself. This suggests that a proper sampling strategy is critical in semi-supervised learning.

1 Introduction

With the dramatic increase in the amount of textual information available in digital archives and the WWW, there has been growing interest in techniques for automatically extracting information from text documents. Information Extraction (IE) is such a technology that IE systems are expected to identify relevant information (usually of pre-defined types) from text documents in a certain domain and put them in a structured format.

According to the scope of the NIST Automatic Content Extraction (ACE) program (ACE, 2000-2007), current research in IE has three main objectives: Entity Detection and Tracking (EDT), Relation Detection and Characterization (RDC), and Event Detection and Characterization (EDC). This paper focuses on the ACE RDC subtask, where many machine learning methods have been proposed, including supervised methods (Miller et al., 2000; Zelenko et al., 2002; Culotta and Soresen, 2004; Kambhatla, 2004; Zhou et al., 2005; Zhang et al., 2006; Qian et al., 2008), semi-supervised methods (Brin, 1998; Agichtein and Gravano, 2000; Zhang, 2004; Chen et al., 2006; Zhou et al., 2008), and unsupervised methods (Hasegawa et al., 2004; Zhang et al., 2005).

Current work on semantic relation extraction task mainly uses supervised learning methods, since it achieves relatively better performance. However this method requires a large amount of manually labeled relation instances, which is both time-consuming and laborious. In the contrast, unsupervised methods do not need definitions of relation types and hand-tagged data, but it is difficult to evaluate their performance since there are no criteria for evaluation. Therefore, semi-supervised learning has received more and more attention, as it can balance the advantages and disadvantages between supervised and unsupervised methods. With the plenitude of unlabeled natural language data at hand, semi-supervised learning can significantly reduce the need for labeled data with only limited sacrifice in performance. Specifically, a bootstrapping algorithm chooses the unlabeled instances with the highest probability of being correctly labeled and use them to augment labeled training data iteratively.

Although previous work (Yarowsky, 1995; Blum and Mitchell, 1998; Abney, 2000; Zhang, 2004) has tackled the bootstrapping approach from both the theoretical and practical point of view, many key problems still remain unresolved, such as the selection of initial seed set. Since the size of the initial seed set is usually small (e.g.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1437–1445,
Singapore, 6-7 August 2009.

100 instances), the imbalance of relation types or manifold structure (cluster structure) in it will severely weaken the strength of bootstrapping. Therefore, it is critical for a bootstrapping approach to select the most appropriate initial seed set. However, current systems (Zhang, 2004; Chen et al., 2006) use a randomly sampling strategy, which fails to explore the affinity nature among the training instances. Alternatively, Zhou et al. (2008) bootstrap a set of weighted support vectors from both labeled and unlabeled data using SVM. Nevertheless, the initial labeled data is still randomly generated only to ensure that there are at least 5 instances for every relation subtype.

This paper presents a new approach to selecting the initial seed set based on stratified sampling strategy in the bootstrapping procedure for semi-supervised semantic relation classification. The motivation behind the stratified sampling is that every relation type should be as much as possible represented in the initial seed set, thus leading to more instances with diverse structures being added to the labeled set. In addition, we also explore different strategies to augment reliably classified instances to the labeled data iteratively, and attempt to find a stoppage criterion for the iteration procedure to greatly decrease the training time, other than using up all the unlabeled set.

The rest of this paper is organized as follows. First, Section 2 reviews related work on semi-supervised relation extraction. Then we present an underlying supervised learner in Section 3. Section 4 details various key aspects of the bootstrapping procedure, including the stratified sampling strategy. Experimental results are reported in Section 5. Finally we conclude our work in Section 6.

2 Related Work

Within the realm of information extraction, currently there are several representative semi-supervised learning systems for extracting relations between named entities.

DIPRE (Dual Iterative Pattern Relation Expansion) (Brin, 1998) is a system based on bootstrapping that exploits the duality between patterns and relations to augment the target relation starting from a small sample. However, it only extracts simple relations such as (*author, title*) pairs from the WWW. Snowball (Agichtein and Gravano, 2000) is another bootstrapping-based system that extracts relations from unstructured text. Snowball shares much in common with DIPRE, including the use of both the bootstrapping framework and the pattern matching approach to extract new unlabeled instances. Due to pattern matching techniques, their systems are hard to be adapted to the general problem of relation extraction.

Zhang (2004) approaches the relation classification problem with bootstrapping on top of SVM. He uses various lexical and syntactic features in the *BootProject* algorithm based on random feature projection to extract top-level relation types in the ACE corpus. Evaluation shows that bootstrapping can alleviate the burden of hand annotations for supervised learning methods to a certain extent.

Chen et al. (2006) investigate a semi-supervised learning algorithm based on label propagation for relation extraction, where labeled and unlabeled examples and their distances are represented as the nodes and the weights of edges respectively in a connected graph, then the label information is propagated from any vertex to nearby vertices through weighted edges iteratively, finally the labels of unlabeled examples are inferred after the propagation process converges.

Zhou et al. (2008) integrate the advantages of SVM bootstrapping in learning critical instances and label propagation in capturing the manifold structure in both the labeled and unlabeled data, by first bootstrapping a moderate number of weighted support vectors through a co-training procedure from all the available data, and then applying label propagation algorithm via the bootstrapped support vectors.

However, in most current systems, the initial seed set is selected randomly such that they may not adequately represent the inherent structure of unseen examples, hence the power of bootstrapping may be severely weakened.

This paper presents a simple yet effective approach to generate the initial seed set by applying the stratified sampling strategy, originated from statistics theory. Furthermore, we try to employ the same stratified strategy to augment the labeled set. Finally, we attempt to find a reasonable criterion to terminate the iteration process.

3 Underlying Supervised Learning

A semi-supervised learning system usually consists of two relevant components: an underlying supervised learner and a

bootstrapping algorithm on top of it. In this section we discuss the former, while the latter will be described in the following section.

In this paper, we select Support Vector Machines (SVMs) as the underlying supervised classifier since it represents the state-of-the-art in the machine learning research community, and there are good implementations of the algorithm available. Specifically, we use LIBSVM (Chang et al., 2001), an effective tool for support vector classification, since it supports multi-class classification and provides probability estimation as well.

For each pair of entity mentions, we extract and compute various lexical and syntactic features, as employed in a state-of-the-art relation extraction system (Zhou et al., 2005).

(1) Words: According to their positions, four categories of words are considered: a) the words of both the mentions; b) the words between the two mentions; c) the words before M1; and d) the words after M2.

(2) Entity type: This category of features concerns about the entity types of both the mentions.

(3) Mention Level: This category of features considers the entity level of both the mentions.

(4) Overlap: This category of features includes the number of other mentions and words between two mentions. Typically, the overlap features are usually combined with other features such as entity type and mention level.

(5) Base phrase chunking: The base phrase chunking is proved to play an important role in semantic relation extraction. Most of the chunking features concern about the headwords of the phrases between the two mentions.

In this paper, we do not employ any deep syntactic or semantic features (such as dependency tree, full parse tree etc.), since they contribute quite limited in relation extraction.

4 Bootstrapping & Stratified Sampling

We first present the self-bootstrapping algorithm, and then discuss several key problems on bootstrapping in the order of initial seed selection, augmentation of labeled data and stoppage criterion for iteration.

4.1 Bootstrapping Algorithm

Following Zhang (2004), we define a basic self-bootstrapping strategy, which keeps augmenting the labeled data set with the models straightforwardly trained from previously available labeled data as follows:

Algorithm self-bootstrapping

Require: labeled seed set L
Require: unlabeled data set U
Require: batch size S
Repeat
  Train a single classifier on L
  Run the classifier on U
  Find at most S instances in U that the classifier has the highest prediction confidence
  Add them into L
Until: no data points available or the stoppage condition is reached

Figure 1. Self-bootstrapping algorithm

In order to measure the confidence of the classifier's prediction, we compute the entropy of the label probability distribution that the classifier assigns to the class label on an example (the lower the entropy, the higher the confidence):

$$H = -\sum_{i}^{n} p_i \log p_i \qquad (1)$$

Where n denotes the total number of relation classes, and p_i denotes the probability of current example being classified as the ith class.

4.2 Stratified Sampling for Initial Seeds

Normally, the number of available labeled instances is quite limited (usually less than 100 instances) when the iterative bootstrapping procedure begins. If the distribution of the initial seed set fails to approximate the distribution of the test data, the augmented data generated from bootstrapping would not capture the essence of relation types, and the performance on the test set will significantly decrease even only after one or two rounds of iterations. Therefore, the selection of initial seed set plays an important role in bootstrapping-based semantic relation extraction.

Sampling is a part of statistical practice concerned with the selection of individual observations, which is intended to yield some knowledge about a population of interest. When dealing with the task of semi-supervised semantic relation classification, the population is the training set of relation instances from the ACE RDC corpora. We compare two practical sampling strategies as follows:

(1) *Randomly sampling*, which picks the initial seeds from the training data using a random scheme. Each element thus has an equal probability of selection, and the population is not

subdivided or partitioned. Currently, most work on semi-supervised relation extraction employs this method. However, since the size of the initial seed set is very small, they are not guaranteed to capture the statistical properties of the whole training data, let alone of the test data.

(2) *Stratified sampling*. When the population embraces a number of distinct categories, *stratified sampling* (Neyman, 1934) can be applied to this case. First, the population can be organized by these categories into separate "strata", then a sample is selected within each "stratum" separately, and randomly. Generally, the sample size is normally proportional to the relative size of the strata. The main motivation for using a stratified sampling design is to ensure that particular groups within a population are adequately represented in the sample.

It is well known that the number of the instances for each relation type in the ACE RDC corpora is greatly unbalanced (Zhou et al., 2005) as shown in Table 1 for the ACE RDC 2004 corpus. When the relation instances for a specific relation type occurs frequently in the initial seed set, the classifier will achieve good performance on this type, otherwise the classifier can hardly recognize them from the test set. In order for every type of relations to be properly represented, the stratified sampling strategy is applied to the seed selection procedure.

Types	Subtypes	Train	Test
PHYS	Located	593	145
	Near	70	17
	Part-Whole	299	79
PER-SOC	Business	134	39
	Family	101	20
	Other	44	11
EMP-ORG	Employ-Executive	388	101
	Employ-Staff	427	112
	Employ-Undetermined	66	12
	Member-of-Group	152	39
	Subsidiary	169	37
	Partner	10	2
	Other	64	16
ART	User-or-Owner	160	40
	Inventor-or-Man.	8	1
	Other	1	1
OTHER-AFF	Ethnic	31	8
	Ideology	39	9
	Other	43	11
GPE-AFF	Citizen-or-Resid.	226	47
	Based-In	165	50
	Other	31	8
DISC		224	55
Total		3445	860

Table 1. Numbers of relations on the ACE RDC 2004: break down by relation types and subtypes

Figure 2 illustrates the stratified sampling strategy we use in bootstrapping, where *RSET* denotes the training set, *V* is the stratification variable, and *SeedSET* denotes the initial seed set. First, we divide the relation instances into different strata according to available properties, such as major relation type (considering reverse relations or not) and relation subtype (considering reverse relations or not). Then within every stratum, a certain number of instances are sampled randomly, and this number is normally proportional to the size of that stratum in the whole population. However, when this number is 0 due to the rounding of real numbers, it is set to 1. Also it must be ensured that the total number of instances being sampled is N_S. Finally, these instances form the initial seed set and can be used as the input to the underlying supervised learning for the bootstrapping procedure.

Require: $RSET = \{R_1, R_2, \ldots, R_N\}$
Require: $V = \{v_1, v_2, \ldots, v_K\}$
Require: *SeedSET* with the size of N_S (100)
Initialization:
$SeedSET = NULL$
Steps:

- Group *RSET* into *K* strata according to the stratified variable *V*, i.e.:
  $RSET = \{RSET_1, RSET_2, \ldots, RSET_K\}$
- Calculate the class prior probability for each stratum i={1,2,…,*K*}
  $P_i = NUM(RSET_i) / NUM(RSET)$
- Caculate the number of intances being sampled for each stratum
  $N_i = P_i * N$
  If $N_i = 0$ then $N_i = 1$
- Calculate the difference of numbers as follows:
  $$N_\Delta = N_S - \sum_{i=1}^{K} N_i$$
- If $N_\triangle > 0$ then add N_i (i=1,2,…,$|N_\triangle|$) by 1
  If $N_\triangle < 0$ then subtract 1 from N_i (i=1,2,...,$|N_\triangle|$)
- For each *i* from 1 to *K*
  Select N_i instances from $REST_i$ randomly
  Add them into *SeedSET*

Figure 2. Stratefied Sampling for initial seeds

4.3 Augmentation of labeled data

After each round of iteration, some newly classified instances with the highest confidence can be augmented to the labeled training data. Nevertheless, just like the selection of initial seed set, we still wish that every stratum would be represented as appropriately as possible in the

instances added to the labeled set. In this paper, we compare two kinds of augmentation strategies available:

(1) *Top n* method: the classified instances are first sorted in the ascending order by their entropies (i.e. decreasing confidence), and then the top *n* (usually 100) instances are chosen to be added.

(2) *Stratified* method: in order to make the added instances representative for their stratum, we first select *m* (usually greater than *n*) instances with the highest confidence, then we choose *n* instances from them using the stratified strategy.

4.4 Stoppage of Iterations

In a self-bootstrapping procedure, as the iterations go on, both the reliable and unreliable instances are added to the labeled data continuously, hence the performance will fluctuate in a relatively small range. The key question here is how we can know when the bootstrapping procedure reaches its best performance on the test data. The bootstrapping algorithm by Zhang (2004) stops after it runs out of all the training instances, which may take a relatively long time. In this paper, we present a method to determine the stoppage criterion based on the mean entropy as follows:

$$H_i <= p \qquad (2)$$

Where H_i denotes the mean entropy of the confidently classified instances being augmented to the labeled data in each iteration, and p denotes a threshold for the mean entropy, which will be fixed through empirical experiments. This criterion is based on the assumption that when the mean entropy becomes less than or equal to a certain threshold, the classifier would achieve the most reliable confidence on the instances being added to the labeled set, and it may be impossible to yield better performance since then. Therefore, the iteration may stop at that reasonable point.

5 Experimentation

This section aims to empirically investigate the effectiveness of the bootstrapping-based semi-supervised learning we discussed above for semantic relation classification. In particular, different methods for selecting the initial seed set and augmenting the labeled data are evaluated.

5.1 Experimental Setting

We use the ACE corpora as the benchmark data, which are gathered from various newspapers, newswire and broadcasts. The ACE 2004 corpus contains 451 documents and 5702 positive relation instances. It defines 7 relation types and 23 subtypes between 7 entity types. For easy reference with related work in the literature, evaluation is also done on 347 documents (including nwire and bnews domains) and 4305 relation instances using 5-fold cross-validation. That is, these relation instances are first divided into 5 sets, then, one of them (about 860 instances) is used as the test data set, while the others are regarded as the training data set, from which the initial seed set is sampled. In the ACE 2003 corpus, the training set consists of 674 documents and 9683 positive relation instances while the test data consists of 97 documents and 1386 positive relation instances. The ACE RDC 2003 task defines 5 relation types and 24 subtypes between 5 entity types.

The corpora are first parsed using Collins's parser (Collins, 2003) with the boundaries of all the entity mentions kept. Then, the parse trees are converted into chunklink format using chunklink.pl [1]. Finally, various useful lexical and syntactic features, as described in Subsection 3.1, are extracted and computed accordingly. For the purpose of comparison, we define our task as the classification of the 5 or 7 major relation types in the ACE RDC 2003 and 2004 corpora.

For LIBSVM parameters, we adopted the polynomial kernel, and *c* is set to 10, *g* is set to 0.15. Under this setting, we achieved the best classification performance.

5.2 Experimental Results

In this subsection, we compare and discuss the experimental results using various sampling strategies, different augmentation methods, and iteration stoppage criterion.

Comparison of sampling strategies in selecting the initial seed set

Table 2 and Table 3 show the initial and the highest classification performance of Precision/Recall/F-measure for various sampling strategies of the initial seed set on 7 major relation types of the ACE RDC 2004 corpus respectively when the size of initial seed set *L* is 100, the batch size *S* is 100, and the top 100

[1] http://ilk.kub.nl/~sabine/chunklink/

instances with the highest confidence are added at each iteration. Table 2 also lists the number of strata for stratified sampling methods from which the initial seeds are randomly chosen respectively. Table 3 additionally lists the time needed to complete the bootstrapping process (on a PC with a Pentium IV 3.0G CPU and 1G memory). In this paper, we consider the following five experimental settings when sampling the initial seeds:

- *Randomly* Sampling: as described in Subsection 4.2.
- *Stratified-M* Sampling: the strata are grouped in terms of major relation types without considering reverse relations.
- *Stratified-MR* Sampling: the strata are grouped in terms of major relation types, including reverse relations.
- *Stratified-S* Sampling: the strata are grouped in terms of relation subtypes without considering reverse subtypes.
- *Stratified-SR* Sampling: the strata are grouped in terms of relation subtypes, including reverse subtypes.

For each sampling strategies, we performed 20 trials and computed average scores and the total time on the test set over these 20 trials.

Sampling strategies for initial seeds	# of strat.	P(%)	R(%)	F
Randomly	1	66.1	65.9	65.9
Stratified-M	7	69.1	66.5	67.7
Stratified-MR	13	69.3	67.3	68.2
Stratified-S	30	69.8	67.7	68.7
Stratified-SR	39	69.9	68.5	69.2

Table 2. The initial performance of applying various sampling strategies to selecting the initial seed set on the ACE RDC 2004 corpus

Sampling strategies for initial seeds	Time (min)	P(%)	R(%)	F
Randomly	52	68.6	66.2	67.3
Stratified-M	65	71.0	66.9	68.8
Stratified-MR	65	71.6	67.0	69.2
Stratified-S	71	72.7	67.8	70.1
Stratified-SR	77	72.9	68.4	70.6

Table 3. The highest performance of applying various sampling strategies in selecting the initial seed set on the ACE RDC 2004 corpus

These two tables jointly indicate that the self-bootstrapping procedure for all sampling strategies can moderately improve the classification performance by ~1.2 units in F-score, which is also verified by Zhang (2004). Furthermore, they show that:

- The most improvements in performance come from improvements in precision. Actually, for some settings the recalls even decrease slightly. The reason may be that due to the nature of self-bootstrapping, the instances augmented at each iteration are always those which are the most similar to the initial seed instances, therefore the models trained from them would exhibit higher precision on the test set, while it virtually does no help for recall.
- All of the four stratified sampling methods outperform the randomly sampling method to various degrees, both in the initial performance and the highest performance. This means that sampling of the initial seed set based on stratification by major/sub relation types can be helpful to relation classification, largely due to the performance improvement of the initial seed set, which is caused by adequate representation of instances for every relation type.
- Of all the four stratified sampling methods, the Stratified-SR sampling achieves the best performance of 72.9/68.4/70.6 in P/R/F. Moreover, the more the number of strata generated by the sampling strategy, the more appropriately they would be represented in the initial seed set, and the better performance it will yield. This also implies that the hierarchy of relation types/subtypes in the ACE RDC 2004 corpus is fairly reasonably defined.
- An important conclusion, which can be draw accordingly, is that the F-score improvement of *Stratified-SR* sampling over Randomly sampling in initial performance (3.3 units) is significantly greater than the F-score improvement gained by bootstrapping itself using *Randomly* sampling (1.4 units). This means that the sampling strategy of the initial seed set is even more important than the bootstrapping algorithm itself for relation classification.
- It is interesting to note that the time needed to bootstrap increases with the number of strata. The reason may be that due to more diverse structures in the labeled data for stratified sampling, the SVM needs more time to differentiate between instances, i.e. more time to learn the models.

Comparison of different augmentation strategies of training data

Figure 3 compares the performance of F-score for two augmentation strategies: the *Top n* method and the *stratified* method, over various initial seed sampling strategies on the ACE RDC 2004 corpus. For each iteration, a variable

number (*m* is ranged from 100 to 500) of classified instances in the decreasing order of confidence are first chosen as the base examples, then at most 100 examples are selected from the base examples to be augmented to the labeled set. Specifically, when *m* is equal to 100, the whole set of the base example is added to the labeled data, i.e. degenerated to the *Top n* augmentation strategy. On the other hand, when *m* is greater than 100, we wish we would select examples of different major relation types from the base examples according to their distribution in the training set, in order to achieve the performance improvement as much as the stratified sampling does in the selection of the initial seed set.

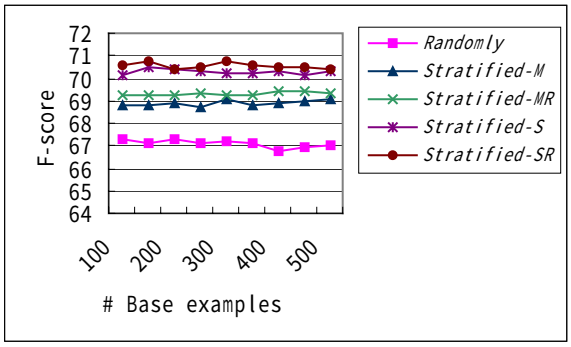


Figure 3. Comparison of two augmentation strategies over different sampling strategies in selecting the initial seed set.

This figure shows that, except for *randomly* sampling strategy, the stratified augmentation strategies improve the performance. Nevertheless, this result is far from our expectation in two ways:

- The performance improvement in F-score is trivial, at most 0.4 units on average. The reason may be that, although we try to add as many as 100 classified instances to the labeled data according to the distribution of every major relation type in the training set, the top *m* instances with the highest confidence are usually focused on certain relation types (e.g. PHSY and PER-SOC), this leads to the stratified augmentation failing to function effectively. Hence, all the following experiments will only adopt *Top n* method for augmenting the labeled data.
- With the increase of the number of the base examples, the performance fluctuates slightly, thus it is relatively difficult to recognize where the optima is. We think there are two contradictory factors that affect the performance. While the reliability of the instances extracted from the base examples decreases with the increase of the number of base examples, the probability of extracting instances of more relation types increases with the increase of the number of the base examples. These two factors inversely interact with each other, leading to the fluctuation in performance.

Comparison of different threshold values for stoppage criterion

We compare the performance and bootstrapping time (20 trials with the same initial seed set) when applying stoppage criterion in Formula (2) with different threshold *p* over various sampling strategies on the ACE RDC 2004 corpus in Figure 4 and Figure 5 respectively. These two figures jointly show that:

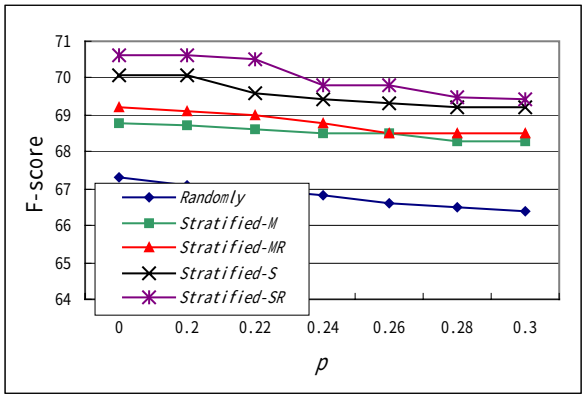


Figure 4. Performance for different *p* values

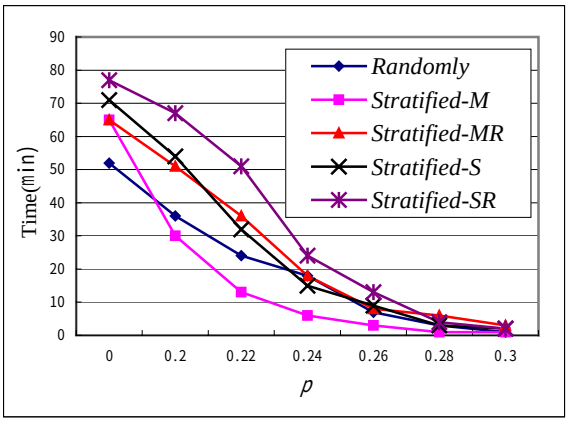


Figure 5. Bootstrapping time for different *p* values

- The performance decreases slowly while the bootstrapping time decreases dramatically with the increase of p from 0 to 0.3. Specifically, when the p equals to 0.3, the bootstrapping time tends to be neglected, while the performance is almost similar to the initial performance. It implies that we can find a reasonable point for each sampling strategy, at which the time falls greatly while the performance nearly does not degrade.

Relation types	*Bootproject*			*LP-js*			*Stratified Bootstrapping*		
	P	R	F	P	R	F	P	R	F
ROLE	78.5	69.7	73.8	81.0	74.7	77.7	74.7	86.3	80.1
PART	65.6	34.1	44.9	70.1	41.6	52.2	66.4	47.0	55.0
AT	61.0	84.8	70.9	74.2	79.1	76.6	74.9	66.1	70.2
NEAR	-	-	-	13.7	12.5	13.0	100.0	2.9	5.6
SOC	47.0	57.4	51.7	45.0	59.1	51.0	65.2	79.0	71.4
Average	67.9	67.4	67.6	73.6	69.4	70.9	73.8	73.3	73.5

Table 4. Comparison of semi-supervised relation classification systems on the ACE RDC 2003 corpus

● Clearly, if the performance is the primary concern, then p=0.2 may be the best choice in that we can get ~30% saving on the time at the cost of only ~0.08 loss in F-score on average. If the time is a primary concern, then p=0.22 is a reasonable threshold in that we get ~50% saving on the time at the cost of ~0.25 units loss in F-score on average. This suggests that our proposed stoppage criterion is effective to terminate the bootstrapping procedure with minor performance loss.

Comparison of *Stratified Bootstrapping* with *Bootproject* and *Label propagation*

Table 4 compares *Bootproject* (Zhang, 2004), *Label propagation* (Chen et al., 2006) with our *Stratified Bootstrapping* on the 5 major types of the ACE RDC 2003 corpus.

Both *Bootproject* and *Label propagation* select 100 initial instances randomly, and at each iteration, the top 100 instances with the highest confidence are added to the labeled data. Differently, we choose 100 initial seeds using stratified sampling strategy; similarly, the top 100 instances with the highest confidence are augmented to the labeled data at each iteration. Due to the lack of comparability followed from the different size of the labeled data used in (Zhou et al., 2008), we omit their results here.

This table shows that our *stratified bootstrapping* procedure significantly outperforms both *Bootproject* and *Label Propagation* methods on the ACE RDC corpus, with the increase of 5.9/4.1 units in F-score on average respectively. *Stratified bootstrapping* consistently outperforms *Bootproject* in every major relation type, while it outperforms *Label Propagation* in three of the major relation types, especially SOC type, with the exception of AT and NEAR types. The reasons may be follows. Although there are many AT relation instances in the corpus, they are scattered divergently in multi-dimension space so that they tend to be relatively difficult to be recognized via SVM. For the NEAR relation instances, they occur least frequently in the whole corpus, so it is very hard for them to be identified via SVM. By contrast, even small size of labeled instances can be fully utilized to correctly induce the unlabeled instances via LP algorithm due to its ability to exploit manifold structures of both labeled and unlabeled instances (Chen et al., 2006).

In general, these results again suggest that the sampling strategy in selecting the initial seed set plays a critical role for relation classification, and stratified sampling can significantly improve the performance due to proper selection of the initial seed set.

6 Conclusion

This paper explores several key issues in semi-supervised learning based on bootstrapping for semantic relation classification. The application of stratified sampling originated from statistics theory to the selection of the initial seed set contributes most to the performance improvement in the bootstrapping procedure. In addition, the more strata the training data is divided into, the better performance will be achieved. However, the augmentation of the labeled data using the stratified strategy fails to function effectively largely due to the unbalanced distribution of the confidently classified instances, rather than the stratified sampling strategy itself. Furthermore, we also propose a mean entropy-based stoppage criterion in the bootstrapping procedure, which can significantly decrease the training time with little loss in performance. Finally, it also shows that our method outperforms other state-of-the-art semi-supervised ones.

Acknowledgments

This research is supported by Project 60673041 and 60873150 under the National Natural Science Foundation of China, Project 2006AA01Z147 under the “863” National High-Tech Research and Development of China,

Project BK2008160 under the Jiangsu Natural Science Foundation of China, and the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20060285008. We would also like to thank the excellent and insightful comments from the three anonymous reviewers.

References

S. Abney. Bootstrapping. 2002. *In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*.

ACE 2002-2007. The Automatic Content Extraction (ACE) Projects. 2007. http//www.ldc.upenn.edu/Projects/ACE/.

E. Agichtein and L. Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the 5th ACM international Conference on Digital Libraries (ACMDL 2000)*.

A. Blum and T. Mitchell. 1996. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the workshop on Computational Learning Theory*. Morgan Kaufmann Publishers.

S. Brin. 1998. Extracting patterns and relations from the world wide web. In *WebDB Workshop at 6th International Conference on Extending Database Technology (EDBT 98)*.

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

M. Collins. 2003. Head-Driven Statistics Models for Natural Language Parsing. *Computational linguistics*, 29(4): 589-617.

J.X. Chen, D.H. Ji, and L.T. Chew. 2006. Relation Extraction using Label Propagation Based Semi supervised Learning. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association of Computational Linguistics (COLING/ACL 2006)*, pages 129-136. July 2006, Sydney, Australia.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association of Computational Linguistics (ACL 2004)*, pages 423-439. 21-26 July 2004, Barcelona, Spain.

T. Hasegawa, S. Sekine, and R. Grishman. 2004. Discovering Relations among Named Entities from Large Corpora. In *Proceedings of the 42nd Annual Meeting of the Association of Computational Linguistics (ACL 2004)*. 21-26 July 2004, Barcelona, Spain.

N. Kambhatla. Combining lexical, syntactic and semantic features with Maximum Entropy models for extracting relations. In *Proceedings of the 42nd Annual Meeting of the Association of Computational Linguistics (ACL 2004)(posters)*, pages 178-181. 21-26 July 2004, Barcelona, Spain.

S. Miller, H. Fox, L. Ramshaw, and R. Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proceedings of the 6th Applied Natural Language Processing Conference*. 29 April-4 May 2000, Seattle, USA.

J. Neyman. 1934. On the Two Different Aspects of the Representative Method: The Method of Stratified Sampling and the Method of Purposive Selection. *Journal of the Royal Statistical Society*, 97(4): 558-625.

L.H. Qian, G.D. Zhou, Q.M. Zhu, and P.D Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of The 22nd International Conference on Computational Linguistics (COLING 2008)*, pages 697-704. 18-22 August 2008, Manchester, UK.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *the Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL 95)*, pages 189-196. 26-30 June 1995, MIT, Cambridge, Massachusetts, USA.

D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel Methods for Relation Extraction. *Journal of Machine Learning Research*, (2): 1083-1106.

M. Zhang, J. Zhang, J. Su, and G.D. Zhou. 2006. A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association of Computational Linguistics (COLING/ACL 2006)*, pages 825-832. Sydney, Australia.

M. Zhang, J. Su, D. M. Wang, G. D. Zhou, and C. L. Tan. 2005. Discovering Relations between Named Entities from a Large Raw Corpus Using Tree Similarity-Based Clustering. In *Proceedings of the 2nd international Joint Conference on Natural Language Processing (IJCNLP-2005)*, pages 378-389. Jeju Island, Korea.

Z. Zhang. 2004. Weakly-supervised relation classification for Information Extraction. In *Proceedings of ACM 13th conference on Information and Knowledge Management (CIKM 2004)*. 8-13 Nov 2004, Washington D.C., USA.

G.D. Zhou, J. Su, J. Zhang, and M. Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association of Computational Linguistics (ACL 2005)*, pages 427-434. Ann Arbor, USA.

G.D. Zhou, J.H. Li, L.H. Qian, and Q.M. Zhu. 2008. Semi-Supervised Learning for Relation Extraction. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP-2008)*, page 32-38. 7-12 January 2008, Hyderabad, India.

Construction of a Blog Emotion Corpus for Chinese Emotional Expression Analysis

Changqin Quan
Faculty of Engineering
University of Tokushima
2-1 Minamijosanjima Tokushima Japan
quan-c@is.tokushima-u.ac.jp

Fuji Ren
Faculty of Engineering
University of Tokushima
2-1 Minamijosanjima Tokushima Japan
ren@is.tokushima-u.ac.jp

Abstract

There is plenty of evidence that emotion analysis has many valuable applications. In this study a blog emotion corpus is constructed for Chinese emotional expression analysis. This corpus contains manual annotation of eight emotional categories (expect, joy, love, surprise, anxiety, sorrow, angry and hate), emotion intensity, emotion holder/target, emotional word/phrase, degree word, negative word, conjunction, rhetoric, punctuation and other linguistic expressions that indicate emotion. Annotation agreement analyses for emotion classes and emotional words and phrases are described. Then, using this corpus, we explore emotion expressions in Chinese and present the analyses on them.

1 Introduction

Textual emotion analysis is becoming increasingly important due to augmented communication via computer mediated communication (CMC) internet sources such as weblogs, email, websites, forums, and chat rooms. Especially, blogspace consists of millions of users who maintain an online diary, containing frequently-updated views and personal remarks about a range of issues.

Despite the increased focus on analysis of web content, there has been limited emotion analysis of web contents, with the majority of studies focusing on sentiment analysis or opinion mining. Classifying the mood of a single text is a hard task; state-of-the-art methods in text classification achieve only modest performance in this domain (Mishne, 2005). In this area, some of the hardest problems involve acquiring basic resources. Corpora are fundamental both for developing sound conceptual analyses and for training these emotion-oriented systems at different levels: to recognize emotions, to express appropriate emotions, to anticipate emotions, and other emotion processing applications.

In this study we propose a relatively fine-grained annotation scheme, annotating emotion in text at three levels: document, paragraph, and sentence. We select eight emotion classes (expect, joy, love, surprise, anxiety, sorrow, angry and hate) for this annotation, and explore various linguistic expressions that indicate emotion in Chinese. The annotation scheme has been employed in the manual annotation of a corpus containing 1,487 documents, with 11,255 paragraphs, 35,096 sentences, and 878,164 Chinese words. Then, using this corpus, we explore and present data analyses on emotions, involving emotion states, accompanying emotions, transfer emotions, independent emotions in texts.

The remainder of this paper is organized as follows. Section 2 describes the emotion corpus annotation scheme. Section 3 presents the inter-annotator agreement study. Section 4 describes the analysis of emotion expressions. Section 5 presents a review of current emotion corpora for textual emotion analysis. Section 6 concludes this study with closing remarks and future directions.

2 Blog Emotion Corpus Annotation Scheme

Weblogs are an increasingly popular mode of communication in the ever changing online world. Writing suits the recording of facts and the communication of ideas, and their textual basis makes them equally suitable for recording emotions and opinions. So, we select blogs as object and data source for this emotion corpus annotation.

2.1 Emotional Expression in Text

An important starting point in constructing this corpus is to represent emotion in text. One of the biggest questions in affect recognition is, "What

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1446–1454,
Singapore, 6-7 August 2009.

are the couplings between affective states and their patterns of expressions" (Picard, 1997).

In this study we propose an emotional expression space model to represent emotion in text, which is hierarchical in consistent with the natural structure of a document. Emotion of a document is represented by a vector

$$\overrightarrow{d} =< e_1, e_2, ..., e_i, ..., e_n > \quad (1)$$

Here, e_i is a basic emotion class contained in document d. The values of e_i range from 0.0 to 1.0 (discrete), indicating the intensities of the basic emotion classes. Similar to a document, emotion of each paragraph and each sentence in a document is represented by an emotion vector.

Basic emotions may be defined in many ways. To decrease confusions on emotion categories' selection and to contain the most common emotion classes in blogs, we select eight emotion classes (expect, joy, love, surprise, anxiety, sorrow, angry and hate) for this manual annotation, and they are agreed by eleven annotators through a testing annotation period. Table 1 shows the numbers of the eight emotion classes in documents, paragraphs, and sentences in this corpus.

Emotions	Doc	Para	Sen
Expect	656	2,145	4,588
Joy	565	2,740	6,211
Love	911	4,991	11,866
Surprise	124	503	1,118
Anxiety	732	4,128	10,115
Sorrow	693	3643	8,166
Angry	189	900	2,221
Hate	335	1,589	3,555
Sum	4,205	20,639	47,840

Table 1: Num. of the eight emotion classes

As shown in Table 1, we have reasonably large counts for all 8 emotions in all 3 units of text. And we also can get the average value for the numbers of emotion classes in each document, each paragraph and each sentence; they are 2.83, 1.84, and 1.36 respectively.

2.2 The Multi-level Annotation Frame

The annotation frame includes 3 levels: document, paragraph, and sentence. Sentence level is the basic level for emotion annotation; the annotation includes intensities of the eight basic emotion classes, emotion holder/target, emotional words/phrases, rhetoric, emotional punctuations, emotion objective/subjective and emotion polarity. Paragraph level is the upper level of sentence level; the annotation includes intensities of the eight basic emotion classes, topic words to reflect the topic of a paragraph, and the number of topic sentence that can express the main points of this paragraph. Document level is the uppermost level; its annotation is similar to paragraph level. The tokenized text files are organized into XML documents. An example document is listed in Figure 1.

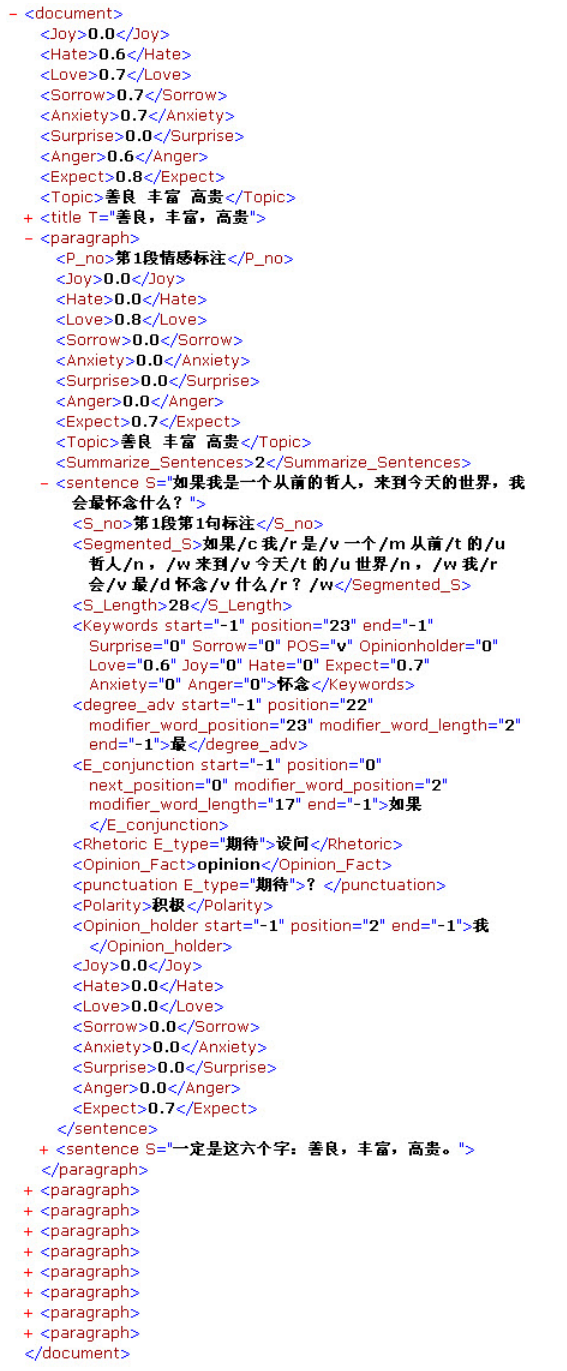


Figure 1: An annotated document in XML format

2.3 Sentence Level Annotation

Sentences are basic units for emotional expression. The central aim of sentence level annotation is to explore as much linguistic expressions for reflecting emotion in Chinese as possible.

a) Emotion holder/target

In the task of opinion analysis, the problem of opinion holder identification has also been studied, (Bethard, Steven et al., 2004; Choi, Cardie, et al., 2005; Kim and Hovy, 2005). As for emotion holder/target identification, little research has been conducted, but we believe it is important for exploring emotional expression and emotion analysis. Emotion holder is the one who holds the emotions, and an emotion target is the object of an emotion holder. For instance,

(1) 我喜欢这个老师。(English: I like this teacher.) In sentence (1),我" (English: I)" is the emotion holder, and 这个老师"(English: this teacher.)" is the emotion target.

In this corpus, not every sentence is annotated with emotion holder or emotion target, and emotion holder or emotion target may not appear in pairs in one sentence. If one sentence has more than one emotion holders or emotion targets, they are all annotated.

b) Emotional words and phrases

Lexicon-based methods have received a lot of attention in opinion analysis task. There are many lexical resources for these tasks. For emotion analysis tasks, the function of words is equally fundamental. In most sentimental lexicons, the words usually bear direct emotions or opinions, such as happy or sad, good or bad. However, there are a lot of sentences can evoke emotions without direct emotional words, for example,

(2) 春天在孩子们的眼里、在孩子们的心里。(English: Spring is in children's eyes, and in their hearts.)

In sentence (2), we may feel joy, love or expect delivered by the writer. Indeed, as (Ortony, Andrew, et al., 1987) indicates, besides words directly referring to emotional states and for which an appropriate lexicon would help, there are words that act only as an indirect reference to emotions depending on the context.

In this annotation scheme, direct emotional words and indirect emotional words in a sentence are all annotated. In sentence (2), 春天"(English: spring)", 孩子们"(English: children)" are labeled. An emotional word or phrase is represented as a vector to record its intensities of the eight basic emotional classes. For instance, the vector for the word 喜欢"(English: like)" $\overrightarrow{w} = (0.0, 0.3, 0.9, 0.0, 0.0, 0.0, 0.0, 0.0)$ indicates the emotions of weak joy and strong love. For indirect emotional words, we annotate their emotion vectors according to their contexts, for example, the possible emotion vector for the word "春天(English: spring)" $\overrightarrow{w} = (0.1, 0.3, 0.3, 0.0, 0.0, 0.0, 0.0, 0.0)$ indicates the emotions of weak expect, joy and love. (The emotions and intensity values may be different because of different annotators).

Emotional phrases are combination of words, such as Chinese proverbs, like "世上无难事，只要肯攀登(English: Where there is a will, there is a way) ". For an emotional phrase, the positions of its first and character in a sentence are labeled, and also for emotional words if there are Chinese word segmentation mistakes.

The statistics show that 84.9% of all emotional words have one emotion, and 14.7% have two emotions, only 0.4% have three or four emotions, but they are indispensable for expressing complex feelings in use of language.

Table 2 shows the numbers of emotional words with different POS (part-of-speech) tags. The set of POS includes 35 classes; Table 2 lists the top five classes.

POS	**Num. of words** (have repeat)
Verb	37,572
Noun	21,308
Adj.	20,265
Adv.	4,223
Gerund	2,789

Table 2: Emotional words with different POS

As shown in Table 2, verbs, nouns, adjectives and adverbs are strong markers of emotion in Chinese.

c) Degree words, negative words, conjunctions

Degree words are associated with the intensities of emotions. In Chinese, degree words appear with high frequency. In this corpus, there are 1,039 different degree words annotated, the total occurring number of them is 16,713, in which, 8,294 degree words modify emotional words or phrases directly. Degree words and the modifying

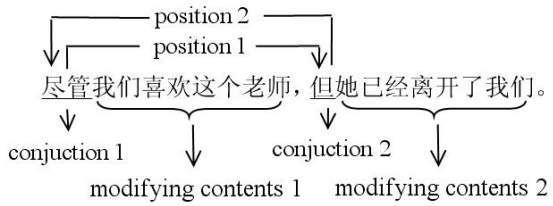
contents are all labeled.

Negative words can be placed almost everywhere in a sentence to change the meaning, also to change the emotions. Negative words are frequently used in Chinese. The statistical data shows that there are 645 different negative words annotated in this corpus, the total occurring number of them is 13,750, in which, 3,668 negative words modify emotional words or phrases directly.

Besides, conjunctions may change the emotion of a sentence. for example,

(3) **尽管**我们喜欢这个老师，**但**她已经离开了我们。(Jin guan wo men xi huan zhe ge lao shi, dan ta yi jing li kai le wo men; English: Although we like this teacher, she has leaved.)

Sentence (3) uses the conjunctions "尽管...但...(jin guan...dan..., English: although)" express emotions of love and sorrow. There are 297 different conjunctions annotated in this corpus. Conjunctions and the modifying contents are all labeled. If conjunctions appear in pairs in a sentence, the position of pairing words for each conjunction are also labeled. For the above sentence (3), conjunctions are annotated as follows (Figure. 2).



English: Although we like this teacher, she has leaved.

Figure 2: An example of conjunctions annotation

Figure 3 shows the growth curve of word number with document number from 300 to 1487. As can be seen from Figure 3, the increase numbers of emotional words/phrases slow down with the increase in the number of documents, and the numbers of negative words, degree words and conjunctions basically remained stable. We can look forward to containing most of common emotional expressions in weblogs articles.

d) Rhetorics, punctuations

Chinese rhetoric has been well studied from the view of linguistics and literature. We select nine common rhetoric categories to annotate: 比喻(English: metaphor), English: 夸张(exaggeration), 拟人(English: personification), 对偶(English: antithesis or parallel), 排比(English: parallelism sentence), 设问(English: rhetorical question with answer), 反问(English: rhetorical question), 重复(English: repeat), 讽刺(English: irony). Especially, 讽刺(English: irony) is a way as to imply the contrary of what one says, if a sentence is annotated with irony, its emotions maybe totally different from the emotions of words that it contains. We annotate rhetoric category and the corresponding emotion category.

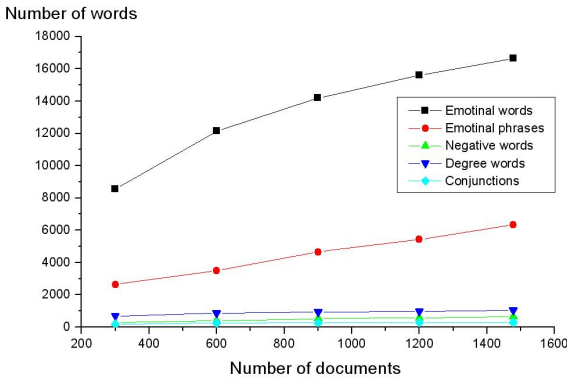


Figure 3: Growth curve of word number

Punctuation is the use of standard marks and signs in writing to separate words into sentences, clauses, and phrases in order to clarify meaning. Some punctuation marks can express emotions, for example, an exclamation mark (!) or a question mark (?) is used at the end of a sentence to show strong emotion. Balog, Mishne, et al. (2006) suggests that people relied on four strategies including punctuation to express happiness versus sadness. Punctuation effect is also shown in (Leshed and Kaye, 2006) to extend to emoticon placement in website text messages. We annotate punctuation with emotion and the corresponding emotion category.

e) Emotion objective/subjective, emotion polarity

Distinguishing a sentence between factual and subjective information could support for many natural language processing applications. Objective and subjective in our annotation scheme is to distinguish a sentence between writer's emotion and non-writer's emotion.

There is a positive side or a negative side on emotion. We call this an emotional polarity. Emotion polarity of a sentence is determined by integrating its emotions. A sentence without emotion is annotated with neutral.

An annotation tool is developed for this corpus

annotation. Input files are text files with Chinese segmentation and part-of-speech tags, the annotated output files are XML files.

3 Annotation Agreement analysis

Emotion annotation is a hard task because the nature of emotion is inherently ambiguous. In the process of annotation, annotators were encouraged to follow their "first intuition". To measure agreement on various aspects of the annotation scheme, three annotators independently annotated 26 documents with a total of 270 paragraphs, 701 sentences.

3.1 Agreement for Emotion Classes

The kappa coefficient of agreement is a statistic adopted by the Computational Linguistics community as a standard measure for this purpose (Carletta, 1996). We measured two agreements for emotion classes' annotation:

Agreement (a): the agreement on classification of containing or not containing some emotions. In this case, we distinguish two classes: emotion intensity $e_i \in \{0.0\}$ or $e_i \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$;

Agreement (b): the agreement on classification of emotion intensity. In this case, we distinguish four classes: $e_i \in \{0.0\}$ or $e_i \in \{0.1, 0.2, 0.3, 0.4\}$, or $e_i \in \{0.5, 0.6, 0.7\}$, or $e_i \in \{0.8, 0.9, 1.0\}$.

Table 3 shows Agreement (a) and (b) measure on documents, paragraphs and sentences.

	Agreement(a)	Agreement(b)
documents	0.831	0.695
paragraphs	0.705	0.616
sentences	0.756	0.648
Average	0.764	0.653

Table 3: Agreement on emotion classes

As shown in Table 3, it is easier for annotators to agree at the coarser levels of granularity, and it is more difficult to agree on the level of emotion intensity.

3.2 Agreement for Emotional Words and Phrases

Measuring agreement for emotional words and phrases is to verify that annotators agree on which expressions should be marked. To illustrate this agreement problem, consider the emotional words and phrases identified by annotators **a** and **b**. This sentence was preprocessed by Chinese segmentation and tagged with part-of-speech.

(4) 今晨/t ， /w 当/d 我/v 沐浴/n 着/u 阳光/n 前往/v 会场/n 时/Ng ， /w 脑中/v 突然/ad 浮现/v 出/v 多年/m 不用/d 的/u 优美/a 词语/n ， /w 那/r 就/d 是/v ： /w 秋高气爽/n 、/w 金光/n 璀璨/z 。(English: This morning, when I walked to the meeting with sunshine, some wonderful words that have not been used for many years crossed my mind, which are "the autumn sky is clear, the air is crisp" and "shinning with gold color")

a: 阳光, 优美, 秋高气爽, 金光, 璀璨;

b: 沐浴, 阳光, 优美, 秋高气爽, 璀璨;

In sentence (4), the two annotators agree that "阳光", "优美", "秋高气爽" and "璀璨" can express emotion. In addition, annotator **a** marked the word "金光", and annotator **b** marked the word "沐浴".

In this task, there is no guarantee that the annotators will identify the same set of expressions. Thus, to measure agreement we want to consider how much intersection there is between the sets of expressions identified by the annotators. We use the following voting-agreement metric to measure agreement in identifying emotional words and phrases.

Metric voting-agreement is defined as follows. Let A, B and C be the sets of expressions annotated by annotators a, b and c respectively. The expert coder is the set of expressions that agreed by at least two annotators, see Equation 2.

$$voting\_agreement = Avg(\frac{count(t_i = e_j)}{count((t_i)}) \quad (2)$$

In which, $t_i \in T, e_j \in E, T = A \bigcup B \bigcup C$, $E = (A \bigcap B) \bigcup (A \bigcap C) \bigcup (B \bigcap C)$.

The agreement for emotional words and phrases is 0.785.

4 Emotional Expressions Analysis

4.1 Emotion State

"Emotion state in text" is the state of combined emotions in a text unit. An emotion state is represented by 8 binary digits, each digit corresponding to a basic emotion class respectively. As an example, a document emotion state "01100000" is the state of combined emotions by joy and love.

The statistics show that, in this corpus, there are 149 different emotion states in all of the 1,487

documents, 165 different emotion states in all of the 11,255 paragraphs, and 143 different emotion states in all of the 35,096 sentences respectively. That indicates the set of emotion state in texts is relatively small. We also found some basic emotions tend to combine together, such as {expect, joy, love}, {anxiety, sorrow}, {angry, hate}. However, some emotions have small or scarce possibility appear together, such as joy and hate, surprise and angry.

4.2 Accompanying Emotions

In an emotion state, some basic emotions are mixed together. When an emotion e_j arise, emotion $e_i (i \neq j)$ arise with accompany, then, e_i is an accompanying emotion of e_j. To compute the probability of the accompanying emotion given an emotion e_j, we count the cooccurrence of e_i and e_j in a text unit (a document, a paragraph, or a sentence).

$$P(e_i|e_j) = \frac{count(e_i \quad with \quad e_j)}{count(e_j)} \tag{3}$$

Table 4 shows the accompanying emotions with the highest probabilities for the eight basic emotions in documents, paragraphs and sentences.

Emotions	Docs	Paras	Sens
Expect	Love	Love	Love
Joy	Love	Love	Love
Love	Joy	Joy	Joy
Surprise	Anxiety	Love	Love
Anxiety	Sorrow	Sorrow	Sorrow
Sorrow	Anxiety	Anxiety	Anxiety
Angry	Anxiety	Hate	Hate
Hate	Anxiety	Sorrow	Angry

Table 4: Accompanying emotions

In Table 4, the accompanying emotions has shown a high uniformity in the 3 units of text.

4.3 Transfer Emotions

When emotion change from one emotion class to another one, we call this emotion transfer. Using the context relation of paragraphs and sentences, we compute the probability $P(e_i \rightarrow e_j)$.

$$P(e_i \rightarrow e_j) = \frac{count(e_t = e_i, \quad e_{t+1} = e_j)}{count(e_t = e_i)} \tag{4}$$

In which, e_t is an emotion class in paragraph t (or sentence t), and e_{t+1} is another emotion class in paragraph $t + 1$ (or sentence $t + 1$). Table 4 shows the transfer emotions with the highest probabilities for the eight basic emotions in paragraphs and sentences.

Emotions	Paras	Sens
Expect	Love	Expect
Joy	Love	Love
Love	Love	Love
Surprise	Love	Love
Anxiety	Anxiety	Anxiety
Sorrow	Sorrow	Sorrow
Angry	Anxiety	Angry
Hate	Hate	Hate

Table 5: Transfer emotions

Similar to this, we can compute the probability of emotion state transfer $P(e\_state_i \rightarrow e\_state_j)$. This may help a lot for emotion prediction, for example, if we know the current emotion state is "00000110" (sorrow an angry), we can estimate the probability of this emotion state to another emotion state "00000001" (hate).

4.4 Independent Emotion

When a text unit (a document, a paragraph, or a sentence) only contains one emotion class, this emotion class is an independent emotion. The statistics show that emotion of love has high independence, however, joy, surprise and angry has relative low independence. The intuition is love can be the only topic emotion in a text unit, but emotions of joy, surprise and anxiety more incline to combine with other emotions.

5 Related work

Previous approaches to textual emotion analysis have employed some different corpora. Mishne (2005) experimented mood classification in blog posts on a corpus of 815,494 blog posts from Livejournal (http://www.livejournal.com), a free weblog service with a large community. Livejournal also used as data source for finding happiness (Mihalcea and Liu, 2006), capturing global mood levels (Mishne and De Rijke, 2006), classifying mood (Jung, Park, et al., 2006; Jung, Choi, et al., 2007), discovering mood irregularities (Balog, Mishne, et al., 2006), recognizing affect (Leshed and Kaye, 2006). A similar emotion corpus in Chinese is Yahoo!'s Chinese news (http://tw.news.yahoo.com), which is used

for Chinese emotion classification of news readers (Lin, Yang, et al., 2007) and emotion lexicon building (Yang, Lin, et al., 2007). Tokuhima (2008) also use web as data resources to obtain a huge collection of emotion-provoking event instances for Japanese emotion classification. More and more weblogs have added mood column to record blog users' moods when they read or write a blog.

Two merits let them well accepted as emotion corpora: a large number of weblogs contained and moods annotated by blog users. However, there is a great inconsistency on emotion categories given by different websites. Livejournal gives a predefined list of 132 common moods, while Yahoo!'s Chinese news provides readers 8 emotion categories. Too many mood classes may confuse users, and Mishne (2005) also pointed out one obvious drawback of the mood "annotation" in this corpora is that they are not provided in a consistent manner; the blog writers differ greatly from each other, and their definitions of moods differ accordingly. In addition, some words are not fitted to be taken as emotion classes, such as "useful" in Yahoo!'s emotion categories. These corpora may be helpful for analyzing the global moods on a full text, but the inconsistent emotion categories is a problem, and no more labeled information can be exploited from them.

The emotion analysis on sentence level may also be important for more detailed emotion analysis systems. Alm, Roth, et al. (2005) explore the text-based emotion prediction problem; they annotated a corpus of 22 Grimms' tales on sentence level with eight emotion categories (angry, disgusted, fearful, happy, sad, positively surprised, negatively surprised), contain 1580 sentences. Neviarouskaya, Prendinger et al. (2007) address the tasks of recognition and interpretation of affect communicated through text messaging. They collected 160 sentences labeled with one of nine emotions categories (anger, disgust, fear, guilt, interest, joy, sadness, shame, and surprise) from a corpus of online diary-like blog posts and a corresponding intensity value. Aman and Szpakowicz (2007) classify emotional and non-emotional sentences based on a knowledge-based approach. They used a corpus with tags of emotion category, emotion intensity and the words/phrases that indicate emotion in text. An emotion corpus for Japanese was built for recognizing emotions and emotion estimation (Ren, 2009; Matsumoto, 2006). However, the sizes of these corpora seem not enough for large scale textual emotion analysis, a lot of linguistic features are not reflected from them. A more fine-grained opinion and emotion corpus is the MPQA Corpus (Wiebe, Wilson, et al., 2005), which contains 535 news articles (10,000-sentence) from a wide variety of news sources, manually annotated at the sentential and subsentential level for opinions and other private states. But emotion categories are not included in it.

To the best of our knowledge, at present, there's no relatively large corpora annotated with detailed linguistic expressions for emotion in Chinese, and we believe that such corpora would support the development and evaluation of emotion analysis systems.

6 Conclusions and Future Work

In this study we proposed an emotional expression space model. Emotion of a document, a paragraph, a sentence, or even a word is represented by an emotional vector. Based on this model, we described a relatively fine-grained annotation scheme and annotated emotion in text. We also gave the inter-annotator agreement study on annotation. Then, we explore the emotional expressions in texts.

This annotated dataset can be obtained for free with license [1]. Eleven annotators made efforts on it spanning a period of ten months (They are Ph.D and M.S. candidates specialize in Natural Language Processing and Emotion Analysis). To ensure the quality of this dataset, each document was performed a three pass annotation, in which the first pass is annotated by one annotator and then the second and the third verification pass were performed by other two annotators. The process of this corpus annotation is easy to make mistakes because of a lot of information should be annotated. The verification pass is to check the annotation mistakes (such as the start and end positions of emotional phrases in sentences), but not to change the choices of emotion classes or emotional words which had been annotated by other annotators.

Using this corpus, we will make a more extensive study of textual emotion analysis in Chinese,

[1] http://a1-www.is.tokushima-u.ac.jp/member/ren/Ren-CECps1.0/Ren-CECps1.0.html

for example, the influence of degree words, negative words, or other elements on emotional expression; the difference between subjective emotion and objective emotion; emotion transfer tracking. More applications also will be explored, such as emotional summarization, emotional question answering; emotional topic discovering. At the same time, new research problems will arise, for examples, how to acquiring more emotional words and to generate their emotional vectors automatically; how to generate emotional vectors for sentences, paragraphs and documents with known emotional elements in them? There is need to immerge further into these problems.

Acknowledgments

We are grateful to our annotators: Huana Li, Ye Wu, Lei Chen, Yu Zhang, Ji Li, Ziliang Du, Yuanlu Fu, Rong Mu, Yan Sun, Cheng Wang, Yunong Wu, and other participants and supporters. We are also grateful to Dr. Suzuki and Dr. Matsumoto for the helpful advice. This research has been partially supported by Ministry of Education, Science, Sprots and Culture, Grant-in-Aid for Challenging Exploratory Research, 21650030.

References

Alena Neviarouskaya, Helmut Prendinger, Mitsuru Ishizuka. 2007. Textual Affect Sensing for Social and Expressive Online Communication. *Proceedings of the 2nd international conference on Affective Computing and Intelligent Interaction*, pp. 218-229.

Bethard, Steven, Hong Yu, Ashley Thornton, Vasileios Hatzivassiloglou, and Dan Jurafsky. 2004. Automatic Extraction of Opinion Propositions and their Holders. *AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications* , pp. 133 – 136.

Changhua Yang, Kevin Hsin-Yih Lin, Hsin-Hsi Chen. 2007. Building Emotion Lexicon from Weblog Corpora. *Proceedings of the ACL 2007 Demo and Poster Sessions*, pp. 133 – 136.

Cecilia Ovesdotter Alm, Dan Roth, Richard Sproat. 2005. Emotions from text: Machine learning for text-based emotion prediction. *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pp. 579-586, Vancouver, British Columbia, Canada.

Fuji Ren. 2009. Affective Information Processing and Recognizing Human Emotion. *Electronic Notes in Theoretical Computer Science,* 225: 39-50.

Gilad Mishne. 2005. Emotions from text: Machine learning for text-based emotion prediction. *Proceedings of Style2005 in SIGIR'05,* pp. 15-19.

Gilad Mishne and Maarten de Rijke. 2006. Capturing global mood levels using blog posts. *AAAI 2006 Spring Symposium on Computational Approaches to Analysing Weblogs,* pp.145-152.

Gilly Leshed and Joseph Kaye. 2006. Understanding how bloggers feel: recognizing affect in blog posts. *Conference on Human Factors in Computing Systems CHI '06 extended abstracts on Human factors in computing systems,* pp. 1019- 1024.

Janyce Wiebe, Theresa Wilson, Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation.* 39: 164 – 210.

Jean Carletta. 1996. Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics.* 22(2):249-254.

Kazuyuki Matsumoto, Fuji Ren, Shingo Kuroiwa. 2006. Emotion Estimation System based on Emotion Occurrence Sentence Pattern. *Computational Intelligence, Lecture Notes in Computer Sciences,* pp.902-911.

Kevin Hsin-Yih Lin, Changhua Yang, Hsin-Hsi Chen. 2007. What emotions do news articles trigger in their readers? *Annual ACM Conference on Research and Development in Information Retrieval,* pp. 733- 734.

Krisztian Balog. Gilad Mishne. Maarten de Rijke. 2006. Why are they excited? identifying and explaining spikes in blog mood levels. *Proceedings 11th Meeting of the European Chapter of the Association for Computational Linguistics,* pp. 207-210.

Ortony, Andrew, Gerald L. Clore, and Mark A. Foss. 1987. The referential structure of the affective lexicon. *Cognitive Science,* 11: 341-364.

Rada Mihalcea and Hugo Liu. 2006. A corpus-based approach to finding happiness. *Proceedings of the AAAI Spring Symposium on Computational,* pp. 139-144.

Rosalind Picard. 1997. *Affective Computing.* The MIT Press, MA, USA.

Ryoko Tokuhisa, Kentaro. Inui, and Yuji. Matsumoto. 2008. Emotion Classification Using Massive Examples Extracted from the Web. *Proceedings of COLING 2008,* pp. 881-888.

Saima Aman and Stan Szpakowicz. 2007. Identifying Expressions of Emotion in Text. *Lecture Notes in Computer Science.* 4629: 196-205.

Soo-Min Kim and Eduard Hovy. 2005. Identifying Opinion Holders for Question Answering in Opinion Texts. *Proceedings of AAAI-05 Workshop on Question Answering in Restricted Domains,* pp. 1367-1373.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying Sources of Opinions with ConditionalRandom Fields and Extraction Patterns. *Proceedings of HLT/EMNLP-05*, pp. 355-362.

Yuchul Jung, Hogun Park, Sung Hyon Myaeng. 2006. A Hybrid Mood Classification Approach for Blog Text. *Lecture Notes in Computer Science,* pp.1099-1103.

Yuchul Jung, Yoonjung Choi, Sung-Hyon Myaeng. 2007. Determining Mood for a Blog by Combining Multiple Sources of Evidence. *IEEE/WIC/ACM International Conference on Web Intelligence,* pp. 271-274.

A Probabilistic Model for Associative Anaphora Resolution

Ryohei Sasano and **Sadao Kurohashi**
Graduate School of Informatics, Kyoto University,
Yoshida-honmachi, Sakyo-ku, Kyoto
{sasano,kuro}@i.kyoto-u.ac.jp

Abstract

This paper proposes a probabilistic model for associative anaphora resolution in Japanese. Associative anaphora is a type of bridging anaphora, in which the anaphor and its antecedent are not coreferent. Our model regards associative anaphora as a kind of zero anaphora and resolves it in the same manner as zero anaphora resolution using automatically acquired lexical knowledge. Experimental results show that our model resolves associative anaphora with good performance and the performance is improved by resolving it simultaneously with zero anaphora.

1 Introduction

The correct interpretation of anaphora is vital for natural language understanding. Bridging anaphora (Clark, 1975) represents a special part of the general problem of anaphora resolution, which has been studied and discussed for various languages and domains (Hahn et al., 1996; Murata et al., 1999; Poesio et al., 2004; Gasperin and Vieira, 2004; Gasperin and Briscoe, 2008).

Usually bridging anaphora considers two types:[1] **associative anaphors** are noun phrases (NPs) that have an antecedent that is necessary to their interpretation but the relation between the anaphor and its antecedent is different from identity; and **indirect anaphors** are those that have an identity relation with their antecedents but the anaphor and its antecedent have different head nouns. In this paper, we focus on associative anaphora in Japanese.

Associative anaphora resolution is decomposed into two steps: acquiring lexical knowledge for associative anaphora resolution, and resolving associative anaphora using the acquired knowledge.

Grammatical salience plays a lesser role for resolving anaphors with full lexical heads, than for pronominal anaphora (Strube and Hahn, 1999; Modjeska, 2002). Furthermore, since associative anaphors and their antecedents usually have different head nouns, string matching technique cannot be applied. Therefore, a large and diverse amount of lexical knowledge is essential to understand associative anaphora. For example, to recognize the meronymic relation between "a house" and "the roof" in (1), such knowledge as "a roof" is a part of a building or vehicle is required. To recognize the attributive relation between "Prius" and "the price" in (2), such knowledge as "price" is a price of some goods or service is required.

(1) There was a house. **The roof** was white.

(2) Toyota launched the hybrid car Prius in 1997. **The price** was 21.5 million yen.

To acquire such lexical knowledge, various studies have been carried out. Early studies used hand-crafted lexical knowledge such as WordNet (Strube and Hahn, 1999; Vieira and Poesio, 2000; Meyer and Dale, 2002), but obtained poor or mediocre results. Hence, Poesio et al. (2002) proposed to exploit "N_h of N_m" phrases in large corpora to resolve associative anaphora in English; Murata et al. (1999) proposed to exploit "N_m *no* N_h" phrases to resolve associative anaphora in Japanese. Here, the Japanese postposition "*no*" roughly corresponds to "of," but it has

[1] The terminology that we use here is introduced by Hawkins (1978), which is also used in (Vieira et al., 2006).

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1455–1464,
Singapore, 6-7 August 2009.

much broader usage. These studies obtained reasonable results, but the coverage of the acquired knowledge was not sufficient. Recently, a number of researchers argued for using the Web as a source of lexical knowledge, and the Web has been shown to be a useful resource for anaphora resolution (Bunescu, 2003; Markert et al., 2003; Poesio et al., 2004).

Hence, in this study, we acquire the lexical knowledge for associative anaphora resolution from "N_m *no* N_h" phrases in the Web by using the method described in (Sasano et al., 2004). We proposed a method for acquiring such lexical knowledge, called *nominal case frames* (NCFs), using an ordinary language dictionary and "N_m *no* N_h" phrases, and constructed NCFs from newspaper corpora. In this study, we aim to acquire a sufficient amount of lexical knowledge by constructing NCFs from the Web.

As for associative anaphora resolution itself, we propose an integrated probabilistic model for zero anaphora and associative anaphora resolution, in which associative anaphora is regarded as a kind of zero anaphora and resolved by using the same model as zero anaphora. Our model assumes zero pronouns that represent indispensable entities of target noun phrases, which are called *zero adnominal* in (Yamura-Takei, 2003), and conducts zero pronoun resolution.

Let us consider the associative anaphoric relation between "*Prius*" and "*kakaku*" (price). Although "*kakaku*" itself is considered as the anaphor from a conventional point of view (3a), our model assumes a zero pronoun ϕ and considers it as the anaphor (3b).

(3) a. *Prius* - *kakaku* (price)
[antecedent: Prius, anaphor: *kakaku* (price)]

b. *Prius* - *(ϕ-no) kakaku* (price (of ϕ))
[antecedent: Prius, anaphor: ϕ]

The point of this study is three-fold: the acquisition of the lexical knowledge for associative anaphora resolution from the Web, the application of zero anaphora resolution model to associative anaphora resolution, and the integrated resolution of zero anaphora and associative anaphora.

2 Construction of Nominal Case Frames

Most nouns have their indispensable entities: "price" is a price of some goods or service, "roof" is a roof of some building, and "coach" is a coach of some sports. The relation between a noun and its indispensable entities is parallel to that between a verb and its arguments or obligatory cases. In this paper, we call indispensable entities of nouns *obligatory cases*. Note that, *obligatory* does not mean grammatically obligatory but obligatory to interpret the meaning of the noun. Associative anaphora resolution needs comprehensive information of obligatory cases of nouns. *Nominal case frames* (NCFs) describe such information, and we construct them from the Web.

2.1 Automatic Construction of NCFs

First, we briefly introduce our method for constructing NCFs from raw corpora proposed in (Sasano et al., 2004).

Whereas verbal case frame construction uses arguments of each verb (Kawahara and Kurohashi, 2002), nominal case frame construction basically uses adnominal constituents of each noun. However, while the meaning of a verbal argument can be distinguished by the postposition, such as "*ga*" (nominative), "*wo*" (accusative), and "*ni*" (dative), the meaning of an adnominal constituent can not be distinguished easily, because most adnominal constituents appear with the same postposition "*no*" (of). Thus, we first conduct a semantic analysis of adnominal constituents, and then construct NCFs using the results as follows:

1. Collect syntactically unambiguous noun phrases "N_m *no* N_h" from the automatic resulting parses of large corpora.
2. Analyze the relation between N_m and N_h by Kurohashi and Sakai's method (1999) that exploits an ordinary language dictionary.
3. Depending on the results, classify N_m, and obtain preliminary case slots for N_h.
4. Merge case slots if two preliminary case slots of N_h are similar.
5. Consider frequent case slots as obligatory cases of N_h. The frequency thresholds are varied according to semantic analyses.
6. For each meaning of N_h, collect case slots and construct case frames.

The point of this method is the integrated use of an ordinary dictionary and example phrases from

Table 1: Examples of constructed nominal case frames.

	Case slot	Examples with freq	Generalized examples with rate
kakaku (1) (price)	Definition: the amount of money you have to pay for **something**.		
	[something]	*shôhin*(goods):9289, *seihin*(product):2520, *buhin*(part):341, *yunyuhin*(importation):232, · · ·	[CT:ARTIFACT]:0.93, · · ·
yane (1) (roof)	Definition: the structure that covers or forms the top of a **building** etc.		
	[building]	*ie*(house):2505, *kuruma*(car):1565, *koya*(hut):895, *tatemono*(building):883,*minka*(private house):679, · · ·	[CT:FACILITY]:0.44, [CT:VEHICLE]:0.13,· · ·
shusho (1) (prime minister)	Definition: the elected leader of the government in a **country** that has a parliament.		
	[country]	*nihon*(Japan):2355, *kuni*(country):272, *doitsu*(Germany):157, *chûgoku*(China):130, · · ·	[NE:LOCATION]:0.82, [CT:VEHICLE]:0.13,· · ·
imouto (1) (sister)	Definition: a girl or woman who has the same parents as you.		
	<relationship>	*watashi*(me):3385, *ore*(me):1188, *boku*(me):898, *jibun*(oneself):341, *tomodachi*(friend):537, · · ·	[CT:PERSON]:0.74, [NE:PERSON]:0.22, · · ·
rebâ(1) (lever)	Definition: a stick or handle on a **machine**.		
	[machine]	*buŕeki*(brake):122, *sokketo*(sochet):67, *waipâ*(wiper):54, *souchi*(device):52,· · ·	[CT:ARTIFACT]:0.61, [CT:VEHICLE]:0.04, · · ·
rebâ(2) (liver)	Definition: the liver of an **animal**, used as food.		
	[animal]	*niwatori*(chicken):153, *buta*(pig):153, *ushi*(cattle):62, *doubutsu*(animal):25,· · ·	[CT:ANIMAL]:0.98, · · ·
senshu(1) (player)	Definition: someone who takes part in a **sport**.		
	[sport]	*yakyû*(baseball):1252, *rirê*(relay):736, *kyôgi*(competition):430, *sakkâ*(soccer):394, · · ·	[CT:ABSTRACTION]:0.56, · · ·
	<affiliation>	*chîmu*(team):4409, *nihon*(Japan):3222, *reddu*(Reds):771, *kankoku*(Korea):644,*rîgu*(league) · · ·	[NE:LOCATION]:0.33, [CT:ORGANIZATION]:0.30, · · ·

* "[]" and "<>" denote dictionary-based and semantic feature-based analysis respectively. For details see (Sasano et al., 2004).

large corpora. Dictionary definition sentences are an informative resource to recognize obligatory cases of nouns. However, it is difficult to resolve associative anaphora by using a dictionary as it is, because all nouns in a definition sentence are not an obligatory case, and only the frequency information of noun phrases tells us which is the obligatory case. On the other hand, a simple method that just collects and clusters "N_m *no* N_h" phrases based on some similarity measure of nouns cannot construct comprehensive nominal case frames, because of polysemy and multiple obligatory cases. For details see (Sasano et al., 2004).

It is desirable to use a probability distribution for deciding whether a case slot is obligatory or not. However, it is difficult to estimate a probability distribution, since we construct nominal case frames not by using the examples of associative anaphora itself but by using the examples of noun phrases "N_m *no* N_h" (N_h of N_m). We use such noun phrases because indispensable entities of noun "N_h" often appear as "N_m." However, we can say neither frequently appeared "N_m" is an indispensable entity of "N_h." nor an indispensable entity frequently appears as "N_m." For example, the name of a country is considered as an indispensable entity of "*shusho*" (prime minister), but does not frequently appear as "N_m."[2] Thus, it is difficult to estimate a probability distribution and we use a hard decision.

2.2 NCF Construction from the Web

We constructed nominal case frames from the Web Corpus (Kawahara and Kurohashi, 2006), which comprises 1.6 billion unique Japanese sentences. In this corpus, there were about 390 million noun phrases "N_m *no* N_h," about 100 million unique noun phrases, and about 17 million unique head nouns "N_h." There were about 4.07 million head nouns that appeared more than 10 times in the corpus, and we used only such head nouns.

The resultant nominal case frames consisted of about 564,000 nouns including compound nouns. We show examples of constructed nominal case frames in Table 1. The average number of case frames for a noun that has case frames was 1.0031, and the average number of case slots for a case frame was 1.0101. However, these statistics differed with the frequency of the noun. Therefore, we investigated the statistics of constructed nominal case frames for each group classified by the frequency of the nouns. Table 2 shows the re-

[2] It is because "the prime minister of Japan" is often mentioned by simply "the prime minister" in Japanese.

Table 2: The statistics of constructed NCFs.

Frequency ranking	Proportion of nouns with NCF	# of NCFs per noun with NCF	# of CSs per NCF	Coverage
-100	56.0%	1.34	1.07	17.3%
-1000	68.8%	1.17	1.16	25.6%
-10000	51.7%	1.11	1.17	27.0%
-100000	14.8%	1.05	1.13	17.6%
100001-	13.7%	1.0009	1.0053	12.5%
all	13.9%	1.0031	1.0101	100%

Table 3: Evaluation of constructed NCFs.

Precision	Recall	F-measure
62/70 (0.89)	62/84 (0.74)	0.81

sult. As for the 10,000 most frequently appeared nouns, which occupied about 70% of all noun appearances, the average number of case frames for a noun was 1.11, and the average number of case slots for a case frame was 1.17.

For evaluating the resultant case frames, we randomly selected 100 nouns from the 10,000 most frequent nouns, and created gold standard case frames for these nouns by hand. For each noun, case frames were given if the noun was considered to have any indispensable entity, and for each case frame, obligatory case slots were given manually: 70 case frames were created that had 84 case slots; 56 case frames had only one case slot, the other 14 case frames had two case slots. 30 nouns had no case frames.

We then evaluated the automatically constructed case slots for these selected nouns. The evaluation result is shown in Table 3: the system output 70 case slots, and out of them, 62 case frames were judged as correct. The F-measure was 0.81. Since the boundary between indispensable cases and optional cases of a noun is not always obvious, this score is considered to be reasonable.

2.3 Generalization of Examples

By using nominal case frames constructed from the Web, sparseness problem was alleviated to some extent, but still remained. For instance, there were thousands of named entities (NEs), which could not be covered intrinsically. To deal with this sparseness problem, we generalized the examples of case slots.

First, we used the categories that Japanese morphological analyzer JUMAN[3] adds to common nouns. In JUMAN, about twenty categories are defined and tagged to common nouns. For example, "*kuruma* (car)," "*niwatori* (chicken)," and "*tatemono* (building)" are tagged as "VEHICLE," "ANIMAL" and "FACILITY," respectively. For each category, we calculated the rate of categorized examples among all case slot examples, and added it to the case slot as "[CT:VEHICLE]:0.13."

We also generalized NEs. We used a common standard NE definition for Japanese provided by IREX workshop (1999). We first recognized NEs in the source corpus by using an NE recognizer (Sasano and Kurohashi, 2008), and then constructed NCFs from the NE-recognized corpus. As well as categories, for each NE class, we calculated the NE rate among all case slot examples, and added it to the case slot as "[NE:PERSON]:0.22." The generalized examples are also included in Table 1.

3 Probabilistic Model

In this study, we apply a lexicalized probabilistic model for zero anaphora resolution proposed in (Sasano et al., 2008) to associative anaphora resolution.

3.1 A Lexicalized Probabilistic Model for Zero Anaphora Resolution

In English, overt pronouns such as "she" and definite noun phrases such as "the company" are anaphors that refer to preceding entities (antecedents). On the other hand, in Japanese, anaphors are often omitted, which are called *zero pronouns*, and zero anaphora resolution is one of the most important techniques for semantic analysis in Japanese.

Here, we introduce our model for zero anaphora resolution (Sasano et al., 2008). This model first resolves coreference and identifies discourse entities; then from the end of each sentence, analyzes each predicate by the following steps:

1. Select a case frame temporarily.
2. Consider all possible correspondences between each input argument and a case slot of the selected case frame.
3. Regard case slots that have no correspondence as zero pronoun candidates.

[3] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/juman-e.html

4. Consider all possible correspondences between zero pronoun candidates and existing entities.

5. For each possible case frame, estimate each correspondence probabilistically, and select the most likely case frame and correspondence.

Figure 1 shows an example of correspondences between case frames and discourse entities.

The probabilistic model gives a probability to each possible case frame CF and case assignment CA when target predicate v, input arguments IA and existing discourse entities ENT are given, and outputs the case frame and case assignment that have the highest probability. That is to say, their model selects the case frame CF_{best} and the case assignment CA_{best} that maximize the probability $P(CF, CA|v, IA, ENT)$:

$$(CF_{best}, CA_{best}) = \underset{CF,CA}{\operatorname{argmax}} P(CF, CA|v, IA, ENT) \quad \text{(i)}$$

By decomposing case assignment (CA) into direct case assignment (DCA) and the indirect case assignment (ICA) and using several independence assumptions, Equation (i) is transformed into the following equation:[4]

$$(CF_{best}, DCA_{best}, ICA_{best}) = \underset{CF,DCA,ICA}{\operatorname{argmax}} \Big(P(CF|v) \times P(DCA, IA|CF) \times P(ICA|ENT, CF, DCA) \Big) \quad \text{(ii)}$$

Here, $P(CF_l|v)$ denotes the probability to select CF_l when target predicate v is given, and estimated by using case structure analysis of large raw corpora.

$P(DCA_k, IA|CF_l)$ denotes the probability to generate direct case assignment and input arguments when a case frame is given, and estimated by using case structure analysis of large raw corpora, the frequency of a case slot example in the automatically constructed verbal case frames, and the web corpus in which the relation between a surface case marker and a case slot is manually annotated.

$P(ICA_k|ENT, CF_l, DCA_k)$ denotes the probability to generate indirect case assignment when existing discourse entities, a case frame and direct case assignments are given, and estimated by using several preferences on the relation between a zero pronoun and an antecedent, such as a lexical preference, a surface case preferences, and a locational preference.

[4]For details see (Sasano et al., 2008).

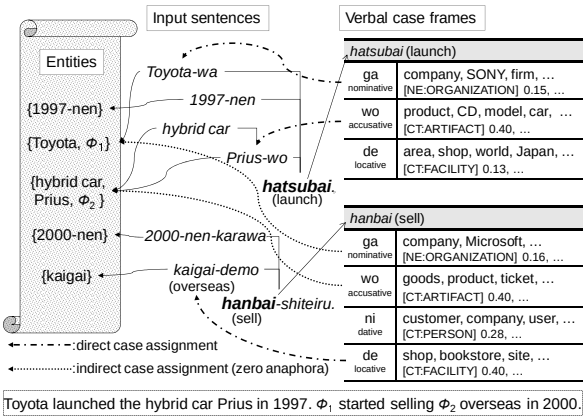


Figure 1: An example of correspondences between verbal case frames and discourse entities.

For example, the lexical preference represents how likely an entity that contains n_{j_m} as a content part is considered to be an antecedent and is estimated by the following equation.

$$\frac{P(n_{j_m}|CF_l, s_j, A'(s_j)=1)}{P(n_{j_m})} \quad \text{(iii)}$$

where, the function $A'(s_j)$ returns 1 if a case slot s_j is filled with an antecedent of a zero pronoun; otherwise 0. $P(n_j|CF_l, s_j, A'(s_j) = 1)$ is calculated by using case frames and denotes the probability of generating a content part n_j of a zero pronoun, when a case frame and a case slot are given and the case slot is filled with an antecedent of a zero pronoun.

3.2 Extension to Associative Anaphora Resolution

We then extend this probabilistic model to associative anaphora resolution. In this model, associative anaphora is regarded as a kind of zero anaphora, that is, the relation between a noun and its obligatory cases is considered to be parallel to that between a verb and its arguments. Omitted obligatory cases are considered to be zero pronouns and resolved by the same process as zero anaphora resolution.

We conduct associative anaphora resolution for only non-coreferent noun phrases. This is because most of the relationships between coreferent noun

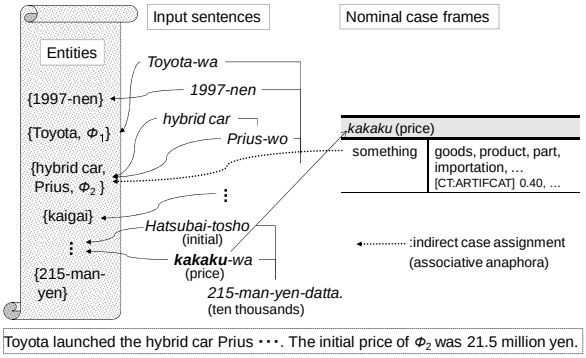


Figure 2: An example of correspondences between a nominal case frame and discourse entities.

phrases and its obligatory entities are easy to recognize by following up the coreference chains. For example, the second appearance of "the roof" in (4) means "the roof of the house," and it is easy to recognize by looking the first appearance of "the roof."

(4) I saw the roof of the house. **The roof** was painted dark green.

While verbal case frames describe both obligatory and optional cases, nominal case frames describe only obligatory cases. Therefore, we consider all case slots of nominal case frames as the target of associative anaphora resolution.

Let us consider following example:

(5) *Toyota-wa 1997-nen hybrid car Prius-wo*
year
hatsubai. 2000-nen-kara-wa kaigai-demo
launched year overseas
hanbai-shiteiru. Hatsubai tosho,
selling initial
(*φ-no*) *kakaku-wa 215-man yen-datta.*
price ten thousands

(Toyota$_1$ launched the hybrid car Prius$_2$ in 1997. ϕ_1 started selling ϕ_2 overseas in 2000. The initial price of ϕ_2 was 21.5 million yen.)

"*Kakaku*" (price) in this example has an omitted obligatory case "[something]" as shown in Table 1. Therefore, our model assumes a zero pronoun and identifies the antecedent from the existing discourse entities, such as {*Toyota*}, {*hybrid-car*, *Prius*},[5] and {*kaigai*}. Figure 2 shows an example of correspondences between the nominal case frame of "*kakaku*" (price) and discourse entities.

In addition, as well as zero anaphora resolution, we exploit generalized examples to estimate lexical preference. When one mention of an entity is tagged any category or recognized as an NE, our model also uses the category or the NE class as the content part of the entity. Specifically, for estimating Equation (iii), our model also calculates:

$$\frac{P(NE{:}ARTIFACT|kakaku(1), no, A'(no){=}1)}{P(NE{:}ARTIFACT)}$$

besides:

$$\frac{P(Prius|kakaku(1), no, A'(no) = 1)}{P(Prius)}$$

and uses the geometric mean of them.

3.3 Salience Score Filtering

Previous work has reported the usefulness of salience for anaphora resolution (Lappin and Leass, 1994; Mitkov et al., 2002). In order to consider the salience of a discourse entity, we introduce the concept of salience score, which is calculated by the following set of simple rules, and only consider the entities that have the salience score no less than 1 as candidate antecedents of an associative anaphor.

- $+2$: mentioned with topical marker "*wa*," or at the end of a sentence.
- $+1$: mentioned without topical marker "*wa*."
- $+1$: assigned to a zero pronoun.
- $\times\alpha$: beginning of each sentence.

We call α a **decay rate**. If $\alpha \geq 1$, we do not filter out any entities. If $\alpha = 0$, we only consider the entities that appears in the same sentence as candidate antecedents. For example, we consider the salience score of the discourse entity {*hybrid-car*, *Prius*} in the example (5) when using $\alpha = 0.6$. In the first sentence, since {*hybrid-car*, *Prius*} is mentioned twice, the salience score is 2.0. At the beginning of the second sentence it becomes 1.2, and after the zero anaphora resolution of "*hanbai*" it becomes 2.2. At the beginning of the third sentence it becomes 1.32.

Note that, this is an ideal case. Practically, some zero pronouns are not detected and some pronouns are assigned wrong antecedent; thus the salience score varies according to the preceding analysis. In addition, the salience score also depends on whether we resolve only associative anaphora or resolve associative anaphora simultaneously with zero anaphora. If zero pronoun resolution is not

[5] "*Hybrid car*" and "*Prius*" are in apposition and these two phrases are considered to refer to the same discourse entity.

conducted, zero pronouns that represent omitted cases of verbs are not considered.

For example, in case of {*hybrid-car*, *Prius*} with $\alpha = 0.6$, if zero anaphora resolution is not conducted, the salience score at the beginning of the third sentence becomes 0.72, because the zero anaphora resolution of "*hanbai*" is not considered; and thus {*hybrid-car*, *Prius*} is not considered as an antecedent candidate.

In order to recognize discourse structure more properly, our model basically resolves associative anaphora simultaneously with zero anaphora, and aims to consider zero pronouns that represent omitted cases of verbs.

3.4 Summary of Our model

Our model is summarized as follows:

1. Parse an input text using the Japanese parser KNP[6] and recognize NEs.

2. Resolve coreference, and link each mention to an entity or create a new entity.

3. From the end of each sentence, zero anaphora and associative anaphora resolution is conducted for each verb and non-coreferent noun by the following steps:

   (a) Select a case frame temporarily.

   (b) Consider all possible correspondences between each input argument and a case slot of the selected case frame.

   (c) Regard case slots that have no correspondence as zero pronoun candidates.

   (d) Consider all possible correspondences between zero pronoun candidates and existing entities that has a salience score no less than 1.0.

   (e) Estimate each correspondence probabilistically, and select the most likely case frame and a correspondence.

4 Experiments

4.1 Setting

We created an anaphoric relation-tagged corpus consisting of 186 web documents (979 sentences), in which all predicate-argument relations and relations between nouns were manually tagged. We show some examples:

[6] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/knp-e.html

(6) *Toyota-wa* *1997-nen* *Prius-wo* *hatsubai.*
year launch
2000-nen-kara-wa *kaigai-demo* *hanbai.*
year overseas sell
(Toyota launched Prius in 1997.
ϕ_1 started selling ϕ_2 overseas in 2000.)

TAG: *hatsubai* $\Leftarrow$ *ga*(NOM):*Toyota*, *wo*(ACC):*Prius*,
hanbai $\Leftarrow$ ***ga*(NOM):*Toyota***, ***wo*(ACC):*Prius***

For the predicate "*hatsubai*" (launch), "*Toyota*" is tagged as *ga* (nominative) case and "*Prius*" is tagged as *wo* (accusative) case. For the predicate "*hanbai*" (sell), "*Toyota*" is tagged as omitted *ga* (nominative) case and "*Prius*" is tagged as omitted *wo* (accusative) case, which are indicated in bold, and such omitted cases are the target of zero anaphora resolution.

As for relations between nouns, both overt and implicit relations are tagged with the Japanese case marker "*no*" (adnominal). In addition, relations between nouns are classified into three categories: indispensable, possible, and adjunct. Since it is not always obvious whether the relations are indispensable or not, borderline relations between indispensable and adjunct are tagged possible. We consider only the implicit relations that are tagged indispensable as the target of associative anaphora resolution.

(7) *Ken-wa* *imouto-to* *yatte-kita.*
sister came.
(Ken came with ϕ's sister.)

TAG: *imouto* $\Leftarrow$ ***no*(ADN):*Ken*** **(indispensable)**

(8) *Kôen-ni* *ikuto* *benchi-ga* *atta.*
park went bench was
(I went to the park. There was a bench in ϕ.)

TAG: *benchi* $\Leftarrow$ *no*(ADN):*kôen* (possible)

We used 62 documents for testing and used the other 124 documents for calculating several probabilities. In the 62 test documents, 110 associative anaphoric relations were tagged. Each parameter for the proposed model was estimated using maximum likelihood from raw corpora, the tagged corpus, and case frames. As verbal case frames, we used the case frames constructed from the Web corpus comprising 1.6 billion sentences (Sasano et al., 2009).

In order to concentrate on associative anaphora resolution, we used the correct morphemes, named entities, syntactic structures, and coreference re-

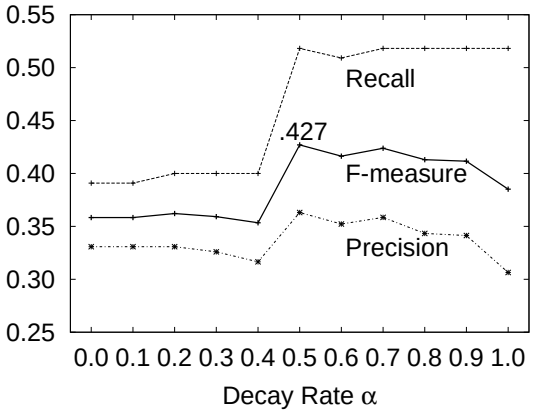


Figure 3: Experimental results of associative anaphora resolution on several salience decay rates α.

lations that were annotated by hand. Since correct coreference relations were given, the number of created entities was the same between the gold standard data and the system output because zero anaphora and associative anaphora resolution did not create new entities.

4.2 Results

Figure 3 shows the experimental results of associative anaphora resolution, in which we used generalized examples, resolved zero anaphora automatically, and varied the decay rate α introduced in Section 3.3 from 0 to 1. When we used the decay rates smaller than 0.5, the recall score worsened clearly. On the other hand, although we expected to obtain higher precision with small decay rate, the highest precision was achieved by the decay rate 0.5. Consequently, we obtained the highest F-measure of 0.427 with the decay rate 0.5. In the following experiments, we fixed the decay rate 0.5.

We utilized two baseline models for demonstrating the effectiveness of our approach: a random model and a salience-based model. The random model selects a case frame and its correspondence randomly from all possible case frames and correspondences. The salience-based model selects a case frame and its correspondence that assign a zero pronoun candidate the existing entity that have highest salience score. In addition, in order to confirm the effectiveness of generalized examples of NCFs, we conducted experiments without using generalized examples. Table 4 shows the experimental results. We can confirm that our proposed model outperforms two baseline models. Without using any generalized examples, the

Table 4: Experimental results of associative anaphora resolution with two baseline models and our model with/without generalized examples.

Model		Recall	Precision	F-measure
Random*		0.148 (16.3/110)	0.035 (16.3/467.5)	0.056
Salience-based		0.400 (44/110)	0.135 (44/325)	0.202
Proposed				
CT	NE			
		0.318 (35/110)	0.257 (35/136)	0.285
	√	0.345 (38/110)	0.268 (38/142)	0.302
√		0.464 (51/110)	0.333 (51/153)	0.388
√	√	**0.518 (57/110)**	**0.363 (57/157)**	**0.427**

CT: Using examples generalized by categories.
NE: Using examples generalized by named entities.
* The average of 10 trials is shown.

F-measure was about 0.14 lower than the method using generalized examples, and we can also confirm the effectiveness of the generalized examples. While generalization of categories much improved the F-measure, generalization of NEs contributed little. This is because the NE rate was smaller than the common noun rate, and so the effect was limited. This tendency was also seen in zero anaphora resolution (Sasano et al., 2008).

In order to investigate the effects of zero anaphora resolution, we tested our model under three conditions: without zero anaphora resolution (no resolution), with zero anaphora resolution (automatically resolved), and with using correct zero anaphora relations that are manually tagged (manually identified). The performance of automatic zero anaphora resolution resulted in a recall of 0.353, a precision of 0.375, and an F-measure of 0.364. Table 5 shows the experimental results. To resolve associative anaphora simultaneously with zero anaphora improved F-measure by 0.072; using correct zero anaphora relations improved F-measure by 0.103. We can confirm that the performance of associative anaphora resolution is improved by considering zero anaphora.

Note that, strictly speaking, these comparisons are not fair because we set the decay rate α to maximize the performance when using generalized examples and resolving zero anaphora automatically. However, these tendencies described above were also seen with other decay rates.

Table 5: The effects of zero anaphora resolution.

Zero anaphora	Recall	Precision	F-measure
No resolution	0.373 (41/110)	0.339 (41/121)	0.355
Automatically resolved	0.518 (57/110)	0.363 (57/157)	0.427
Manually identified	0.573 (63/110)	0.382 (63/165)	0.458

Table 6: Causes of false negatives.

Causes	Num	
Filtered by salience score	22	(15)
Judge as non-anaphoric	13	(14)
Select false antecedents	13	(13)
Case frame sparseness	5	(5)
Total	53	(47)

*"()" denotes the number of causes when using correct zero anaphora tags.

4.3 Discussion

By using generalized examples and resolving simultaneously with zero anaphora, our model achieved a recall of 0.518 (57/110), but there were still 53 associative anaphoric relations that were not recognized. Table 6 shows the causes of them.

22 false negatives were caused by salience score filtering. Note that, it does not mean that these 22 associative anaphoric relations were always recognized correctly if the correct antecedents were not filtered by salience score.

Case frame sparseness caused only 5 false negatives. Considering that the recall of nominal case frames was 74% as shown in Table 3, this seems to be too few. This is because we do not considered the relations that tagged possible, and only considered obviously indispensable relations. From this result, we can say that coverage of nominal case frames for nouns that have obviously indispensable entities is much higher than 74%, which is considered to achieve a coverage of about 95% (105/110).

4.4 Comparison with previous work

Murata et al. (1999) proposed a method of utilizing "N_m *no* N_h" phrases for associative anaphora resolution.[7] They basically used all "N_m *no* N_h" phrases from corpora as a lexical knowledge, and used rule-based approach. They obtained a recall of 0.63 and a precision of 0.68 by using examples of "*X no Y*" (Y of X), a recall of 0.71 and a precision of 0.82 by assuming ideal nominal case frames. One reason of such high performance may be that they considered referential properties of noun phrases, such as generic, indefinite, and definite, while our model does not. We can also say that their experiments were conducted on small and supposedly easy corpora. Half of their corpora were occupied by fairy tale, against which domain specific rules are considered to be effective.

We proposed a rule-based approach for associative anaphora resolution based on automatically acquired nominal case frames (Sasano et al., 2004).[7] We obtained a recall of 0.633 and a precision of 0.508 against news paper articles. However, we regarded some additional relations that can be interpreted by considering coreference relations as associative anaphoric relations.

(9) *Chechen Kyôwakoku-no* ***shuto****-ni ...*
Chechen Republic capital

... ***shuto*** *seiatsu-no saishu dankai-ni ...*
capital conquer last stage

(... to **the capital** of Chechen Republic ... in the last stage to conquer **the capital** ...)

For example, although the second mention of "*shuto*" (capital) in example (9) means "*Chechen Kyôwakoku-no shuto*" (the capital of Chechen Republic), it can be interpreted by recognizing the coreference relation between the first and second mentions of "*shuto*" (capital). Therefore, as mentioned in Section 3.2, we do not consider such relations as associative anaphora in this study; we included such relations as associative anaphora in (Sasano et al., 2004). The relatively high score is caused by this criterion.

5 Conclusion

In this paper, we proposed a probabilistic model for associative anaphora resolution. Our model regards associative anaphora as a kind of zero anaphora and resolves it in the same manner as zero anaphora resolution that uses automatically acquired case frames. We also showed that the performance of associative anaphora resolution can be improved by resolving it simultaneously with zero anaphora. As future work, we plan to consider referential properties of noun phrases in associative anaphora resolution.

[7] Murata et al. (1999) and we (Sasano et al., 2004) used the terminology *indirect anaphora*, but concerned with the same phenomena as we concerned with in this paper.

References

Razvan Bunescu. 2003. Associative anaphora resolution: A web-based approach. In *Proc. of EACL'03: Workshop on The Computational Treatment of Anaphora*, pages 47–52.

Herbert H Clark. 1975. Bridging. In *Proc. of the Conference on Theoretical Issues in Natural Language Processing*, pages 169–174.

Caroline Gasperin and Ted Briscoe. 2008. Statistical anaphora resolution in biomedical texts. In *Proc. of COLING'08*, pages 257–264.

Caroline Gasperin and Renata Vieira. 2004. Using word similarity lists for resolving indirect anaphora. In *Proc. of ACL'04: Workshop on Reference Resolution and its Applications*, pages 40–46.

Udo Hahn, Michael Strube, and Katja Markert. 1996. Bridging textual ellipsis. In *Proc. of COLING'96*, pages 496–501.

John A. Hawkins. 1978. *Definiteness and indefiniteness: a study in reference and grammaticality prediction*. Croom Helm Ltd.

IREX Committee, editor. 1999. *Proc. of the IREX Workshop*.

Daisuke Kawahara and Sadao Kurohashi. 2002. Fertilization of case frame dictionary for robust Japanese case analysis. In *Proc. of COLING'02*, pages 425–431.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case frame compilation from the web using high-performance computing. In *Proc. of LREC'06*, pages 1344–1347.

Sadao Kurohashi and Yasuyuki Sakai. 1999. Semantic analysis of Japanese noun phrases: A new approach to dictionary-based understanding. In *Proc. of ACL'99*, pages 481–488.

Shalom Lappin and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–562.

Katja Markert, Malvina Nissim, and Natalia N Modjeska. 2003. Using the web for nominal anaphora resolution. In *Proc. of EACL'03: Workshop on the Computational Treatment of Anaphora*, pages 39–46.

Josef Meyer and Robert Dale. 2002. Using the WordNet hierarchy for associative anaphora resolution. In *Proc. of SemaNet'02: Building and Using Semantic Networks*.

Ruslan Mitkov, Richard Evans, and Constantin Orăsan. 2002. A new, fully automatic version of Mitkov's knowledge-poor pronoun resolution method. In *Proc. of CICLing'02*.

Natalia N Modjeska. 2002. Lexical and grammatical role constraints in resolution other-anaphora. In *Proc. of DAARC'02*.

Masaki Murata, Hitoshi Isahara, and Makoto Nagao. 1999. Resolution of indirect anaphora in Japanese sentences using examples "X no Y"(Y of X). In *Proc. of ACL'99: Workshop on Coreference and Its Applications*.

Massimo Poesio, Tomonori Ishikawa, Sabine Schulte im Walde, and Renata Vieira. 2002. Acquiring lexical knowledge for anaphora resolution. In *Proc. of LREC'02*, pages 1220–1224.

Massimo Poesio, Pahul Mehta, Axel Maroudas, and Janet Hitzeman. 2004. Learning to Resolve Bridging References. In *Proc. of ACL'04*, pages 143–150.

Ryohei Sasano and Sadao Kurohashi. 2008. Japanese named entity recognition using structural natural language processing. In *Proc. of IJCNLP'08*, pages 607–612.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2004. Automatic construction of nominal case frames and its application to indirect anaphora resolution. In *Proc. of COLING'04*, pages 1201–1207.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2008. A fully-lexicalized probabilistic model for japanese zero anaphora resolution. In *Proc. of COLING'08*, pages 769–776.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2009. The effect of corpus size on case frame acquisition for discourse analysis. In *Proc. of NAACL-HLT'09*, pages 521–529.

Michael Strube and Udo Hahn. 1999. Functional centering – grounding referential coherence in information structure. *Computational Linguistics*, 25(3):309–344.

Renata Vieira and Massimo Poesio. 2000. An empirically based system for processing definite descriptions. *Computational Linguistics*, 26(4):539–592.

Renata Vieira, Eckhard Bick, Jorge Coelho, Vinicius Muller, Sandra Collovini, Jose Souza, and Lucia Rino. 2006. Semantic tagging for resolution of indirect anaphora. In *Proc. of the 7th SIGdial Workshop on Discourse and Dialogue*, pages 76–79.

Mitsuko Yamura-Takei. 2003. Approaches to zero adnominal recognition. In *Proc. of ACL'03: Student Research Workshop*, pages 87–94.

Quantifier Scope Disambiguation Using Extracted Pragmatic Knowledge: Preliminary Results

Prakash Srinivasan
Temple University
1805 N. Broad St.
Wachman Hall 324
Philadelphia, PA 19122
prakash.srinivasan@temple.edu

Alexander Yates
Temple University
1805 N. Broad St.
Wachman Hall 324
Philadelphia, PA 19122
yates@temple.edu

Abstract

It is well known that pragmatic knowledge is useful and necessary in many difficult language processing tasks, but because this knowledge is difficult to acquire and process automatically, it is rarely used. We present an open information extraction technique for automatically extracting a particular kind of pragmatic knowledge from text, and we show how to integrate the knowledge into a Markov Logic Network model for quantifier scope disambiguation. Our model improves quantifier scope judgments in experiments.

1 Introduction

It has long been a goal of the natural language processing (NLP) community to be able to interpret language utterances into logical representations of their meaning. Quantifier scope ambiguity has been recognized as one particularly challenging aspect of this problem. For example, the following sentence has two possible readings, depending on the scope of its quantifiers:

Every boy wants a dog.

One reading of this sentence is that there exists a single dog in the world which all boys want. The second, and usually preferred, reading is that the sentence is describing a separate "wanting" relation for each boy, and that the dog in question is a function of the boy who wants it. In this reading, there may be as many different dogs as boys, although it leaves open the possibility that several of the boys want the same dog. In logic, these two readings can be represented as follows:

1. $\exists_{d \in Dogs} \ \forall_{b \in Boys}$ wants(b, d)
2. $\forall_{b \in Boys} \ \exists_{d \in Dogs}$ wants(b, d)

The readings differ only in the order of the quantifiers. The quantifier that comes first in each expression is said to have *wide scope*; the second quantifier has *narrow scope*.

Linguists and NLP researchers have come up with several theories and mechanisms for automatically determining the scope of quantified linguistic expressions. Despite a long history of proposed solutions, however, researchers have for the most part abandoned this task as hopeless because of "overwhelming evidence suggesting that quantifier scope is a phenomenon that must be treated at the pragmatic level" (Saba and Corriveau, 2001). For example, in active voice clauses, the quantifier for the subject noun is usually preferred for wide scope over the quantifier of the predicate noun (Kurtzman and MacDonald, 1993). But such preferences can easily be overruled by world knowledge:

A doctor lives in every city.

1. $\exists_{d \in Docs} \ \forall_{c \in Cities}$ lives in(d, c)
   (A single doctor lives in all cities.)
2. $\forall_{c \in Cities} \ \exists_{d \in Docs}$ lives in(d, c)
   (Each city has a different doctor living there.)

Syntactic preferences would normally indicate that reading 1 is better, but in this particular case common-sense knowledge of the world overrules that preference and makes reading 2 far more probable.

Open-domain pragmatic knowledge is usually not available to language processing systems, but that is beginning to change. Recent research in open information extraction (Banko and Etzioni, 2008; Davidov and Rappaport, 2008) has shown that we can extract large amounts of relational data from open-domain text with high accuracy. Here, we show how we can connect the two fields, by extracting a targeted form of pragmatic knowledge for use in quantifier scope disambiguation. Our contributions are:

1) We build an extraction mechanism for extracting pragmatic knowledge about relations. In par-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1465–1474,
Singapore, 6-7 August 2009.

ticular, we extract knowledge about the expected sizes of the sets of objects that participate in the relations. The task of identifying functional relationships is a subtask of our extraction problem that has received recent attention in the literature (Ritter et al., 2008).
2) We devise a novel probabilistic model in the Markov Logic Network framework for reasoning over possible readings of sentences that involve quantifier scope ambiguities. The model is able to assign a probability that a particular reading is plausible, given the pragmatic knowledge we extract.
3) We provide an empirical demonstration that our system is able to resolve quantifier scope ambiguities in cases where the syntactic and lexical features used by previous systems are of no help.

The remainder of this paper is organized as follows. The next section describes previous work. Section 3 shows how the problem can be formulated as a task of assigning probabilities to possible worlds, and that the crucial difference between them has to do with the number of objects participating in individual relationships. Section 4 discusses our techniques for extracting the pragmatic knowledge that allows us to make judgments about quantifier scope. Section 5 presents our probabilistic model for resolving scope ambiguities. We present an empirical study in section 6, and section 7 concludes and suggests items for future work.

2 Related Work

Quantifier scope disambiguation has received attention in linguistics and computational linguistics since at least the 1970s. Montague (1973) gave a seminal treatment of quantifier ambiguities, and argued that a particular syntax-based mechanism known as "quantifying-in" could resolve scope ambiguities. Since then, most work on disambiguation has focused on syntactic clues for determining which readings of an ambiguous statement are possible, and of the set of possible readings, which ones are preferred (Van Lehn, 1978; Hobbs and Shieber, 1987; Poesio, 1993a; Hurum, 1988; Moran, 1988). For instance, one linguistic study (Kurtzman and MacDonald, 1993) determined that in active voice sentences where quantifiers in the subject and object give rise to scope ambiguity, there is a preference for the reading in which the subject quantifier has wide scope — the direct reading is acceptable 70-80% of the time, whereas the indirect reading is acceptable 30-40% of the time. Sentences that are similar in all respects except that they are passive voice have no such preference. Nevertheless, in these studies both readings are often quite plausible. In addition to syntactic clues, other studies have noted that the choice of quantifier has a significant effect on scope disambiguation (*e.g.*, "each" has a greater tendency for wide scope than "every") (Van Lehn, 1978; Alshawi, 1990). Most authors have noted that both syntactic and lexical evidence fall short of a full solution, and that pragmatic knowledge (knowledge about the world) is necessary for this task (Van Lehn, 1978; Saba and Corriveau, 1997; Moran, 1988). Saba and Corriveau (2001) recently proposed a test for quantifier scope disambiguation using pragmatic knowledge. However, they do not show how to extract the necessary information, nor do they implement or evaluate their proposed test.

Due to the difficulty of the problem, several authors have devised techniques for "underspecified" logical representations that can efficiently store multiple ambiguous readings, and they devise techniques for automated reasoning using underspecified representations (Reyle, 1995; Latecki, 1992; Poesio, 1993b). Others (Hobbs and Shieber, 1987; Park, 1988) have devised computational mechanisms for generating all of the possible readings of statements exhibiting quantifier ambiguity, especially in cases involving more than two quantifiers.

Detecting functions in extracted relational data has been studied in several contexts. Ritter *et al.*(2008) use knowledge of functions to determine when two extracted relationships contradict one another. Knowledge of functions has also been important in finding synonyms (Yates and Etzioni, 2009) and in review mining (Popescu, 2007). We extend this work by extracting not just a binary determination of whether a relation is functional, but a distribution over the expected number of arguments for that relation. Our technique also differs from previous work based on extracted relationships between named entities. We leverage domain-independent extraction patterns involving numeric phrases, as discussed below; our technique is complementary to existing approaches and could in fact be combined with them for even greater accuracy. Finally, we apply the extracted knowledge in a novel way to quantifier scope disambiguation.

Our work is similar in spirit to several recent

projects that use semantic reasoning over extracted knowledge for a novel approach to well-known tasks. For example, Schoenmackers *et al.*(2008) have recently used extracted knowledge for the task of predicting whether a new extracted fact is correct. Yates *et al.*(2006) use extracted knowledge to determine whether a parse of a sentence has a plausible semantic interpretation. We extend this new line of attack to a hard problem in language understanding.

3 Possible Worlds Framework

We now present a framework for reasoning about quantifier scope ambiguities, and for choosing among possible readings based on pragmatic knowledge (or world knowledge — we use the terms interchangeably). We first present a formal description of the quantifier scope disambiguation (QSD) problem. We then describe the crucial differences between the "possible worlds" evoked by different readings of an ambiguous statement.

3.1 Representation of Readings

We follow Copestake *et al.* (2005), among others, in representing quantifiers as modal operators with three arguments: a variable name for the variable being quantified; a logical formula, called the *restriction*, which defines the set of objects over which the variable may range; and a second logical formula, called the *body*, which defines the expression in which the quantified variable takes part. For example, we represent the sentence "Every dog barks" as: $\texttt{every}(x, \texttt{dog}(x), \texttt{barks}(x))$.

For the sake of clarity and convenience, we restrict our attention to a common syntactic form of sentences, where the semantic representation is relatively well-understood: active-voice English sentences in which the subject noun phrase is quantified, and a noun phrase in the predicate (either an object of the verb, or an object of a preposition attached to the verb) is also quantified. For a sentence with the following structure, in which p_i and q_j represent predicates introduced by modifiers like adjectives and prepositional phrases,

$$(_S\ (_{NP}\ (_{DET}\ Q_1)(_{\overline{N}}\ [p_1, \ldots, p_n]C_1))\\ (_{VP}\ (_V\ R)(_{NP}\ (_{DET}\ Q_2)(_{\overline{N}}\ [q_1, \ldots, q_m]C_2))$$

we can represent the two possible readings of the sentence as:

direct reading:

$$Q_1(x, C_1(x) \wedge p_1(x) \wedge \ldots \wedge p_n(x), \\ Q_2(y, C_2(y) \wedge q_1(y) \wedge \ldots \wedge q_m(y), \\ R(x, y))) \quad (1)$$

indirect reading:

$$Q_2(y, C_2(y) \wedge q_1(y) \wedge \ldots \wedge q_m(y), \\ Q_1(x, C_1(x) \wedge p_1(x) \wedge \ldots \wedge p_n(x), \\ R(x, y))) \quad (2)$$

By making the restriction to this type of sentences, we can isolate the effects of pragmatics on scope disambiguation decisions from the effects of syntax, since all test cases have essentially the same syntax. As we show below, for certain types of relations, the preference for interpretations may be drastically different from the general preference for the direct reading, even though the syntax of the sentences we investigate matches the syntax studied by Kurtzman and MacDonald (1993).

3.2 Readings, Possible Worlds, and World Knowledge

The different logical forms for the direct and indirect readings describe different "possible worlds." For instance, the direct reading of "A doctor lives in every city" describes worlds in which there is a single doctor who manages to reside in each city of the world simultaneously. This reading is "possible" in the sense that it does not contradict itself. In logical terms, if ϕ represents the direct reading of this sentence, $\phi \nvdash \bot$. Using some imagination one could devise a scenario, perhaps in an online game world, that satisfies ϕ.

Nevertheless, the indirect reading is strongly preferred for this statement in the absence of any context that indicates an abnormal world. The indirect reading ϕ' describes worlds where every city is inhabited by some doctor, but potentially a different doctor per city. Using pragmatic knowledge, the reader can easily deduce that this logical statement is a much more likely reading than ϕ. Let B represent the reader's pragmatic knowledge, including facts like "People don't simultaneously live in more than one city," and, "There are at least hundreds of cities in the world." The reader can easily deduce that $B \models \neg\phi$. We now turn to methods for extracting the necessary pragmatic knowledge B from text.

4 Extraction Techniques to Support QSD Decisions

Saba and Corriveau (2001) point out that there is a restricted form of pragmatic knowledge that can be used in many instances of QSD. Consider the facts that were used above to determine that ϕ' is preferable to ϕ. The facts fall into two basic categories of knowledge: 1) the size of class C (*e.g.*, how many cities are there?), and 2) the expected number of Y participants in a relationship R, given that there is exactly 1 X participant (*e.g.*, how many cities does 1 doctor live in?). In both cases, we are concerned with extracting sizes of sets.

Previous extraction systems have attempted to estimate set sizes based on extracted named entities. Downey *et al.* (2005)estimate the size of classes based on the number of named-entities extracted for the class. As far as we are aware, finding the expected size of an argument set for a relation is a novel task for information extraction, but several researchers (Ritter et al., 2008; Yates and Etzioni, 2009) have investigated the special case of detecting functional relations — those relations where the expected size of the Y argument set is precisely 1. As with class size extraction, they use extractions involving named-entity arguments to find functional relations.

Approaches that depend on named-entity extractions have several disadvantages: they must find a large set of named-entities for every set, which can be time-consuming and difficult. Also, many classes, like "trees" and "hot dogs," have no or very few named instances, but many un-named instances, so approaches based on named entities have little hope. In fact, besides classes like people, locations, and organizations (and their subclasses), there are few classes that have a large number of named instances. For classes that do have named instances, synonymy, polysemy, and extraction errors are common problems that can all affect estimates of size (Ritter et al., 2008).

Rather than indirectly determining set sizes from extracted instances, our system directly extracts estimates of set sizes. It uses *numeric phrases*, like "two trees," "hundreds of students," or "billions of stars," to associate numeric values with sets. Table 1 lists the numeric phrases we use. Currently, we use only numeric phrases with explicit values or ranges of values, but it may be possible to increase the recall of our extraction technique by incorporating more approximate phrases like "several," "many," or even bare plurals. We do not match numbers expressed in digits (*e.g.*, 1234) because we found that they produced too many noisy extractions, such as dates and times. For words like "hundreds," we set the value of the word to be the lower limit (*i.e.*, 100). This gives a conservative estimate of the value, but our techniques described below can help to compensate for this bias.

Numeric Phrase	Value
no | none | zero	0
a | one | this | the	1
two	2
⋮	⋮
one hundred | a hundred	100
⋮	⋮
hundreds of	100
thousands of	1,000
tens of thousands of	10,000
⋮	⋮

Table 1: **Numeric phrases used in our extraction patterns.** For the word "the", we require that it be followed directly by a singular noun, to try to weed out plural usages.

Table 2 lists examples of the hand-crafted, domain-independent extraction patterns we use. Our extraction patterns generate two types of extractions, one for classes and one for relations. For classes, each extraction E consists of a class name E_c and a number E_n indicating the size of some subset $S \subseteq E_c$. For instance, the 4gram "hundreds of students are" matches our first pattern. The numeric phrase "hundreds of" here indicates that some subset $S \subseteq E_c =$ students has a size in the hundreds. After processing a large corpus, our system can determine a probability distribution for the size of a class given by:

$$P_C\left(\texttt{size}(C) = N\right) = \frac{|\{E \mid E_c = C \wedge E_n = N\}|}{|\{E \mid E_c = C\}|}$$

In practice, we only include the largest 20% of the numbers N in the set of extractions for a class to estimate that class's size.

The second type of extraction we get from our patterns are relational extractions. Each relational extraction F consists of a relation name F_r, and possibly names for the classes of its two arguments, F_{c1}, F_{c2}. In addition, the extraction contains values for the size of both arguments, F_{n1}

Pattern	Extraction
<numeric> <word>+ (`of` | `are` | `have`)	$E_c =$ <word>+, $E_n = value($<numeric>$)$
(`I` | `he` | `she`) <word>+ <numeric> <noun>	$F_r =$ <word>+, $F_{c1} =$ people, $F_{c2} =$ <noun>, $F_{n1} = 1, F_{n2} = value($<numeric>$)$
`it is` *pastParticiple*(<verb>) `by` <numeric>	$F_r =$ <verb>, $F_{c2} =$ thing, $F_{n1} = value($<numeric>$), F_{n2} = 1$
`is the` <word> `of` <numeric>	$F_r =$ `is the` <word> `of`, $F_{n1} = 1, F_{n2} = value($<numeric>$)$

Table 2: **Sample extraction patterns for discovering classes (E_c) and their sizes (E_n), or relations (F_r) and the expected set size of their arguments (F_{n1} and F_{n2}).**

and F_{n2} respectively. For example, the fragment "she visited four countries" matches the second pattern in Table 2, with $F_r =$ visited, $F_{n1} = 1$, and $F_{n2} = 4$. Note that in our extraction patterns, one of of the arguments is always constrained to be a singleton set, like "he" or "it." This restriction allows us to avoid quantifier scope ambiguity in the extraction process: if we extracted phrases like "Two men married two women," it would be unclear which quantifier has wide scope, and therefore how many men and women are participating in each "married" relationship. By using singular pronouns, we avoid this confusion; in almost all cases, these pronouns have wide scope, and indicate a single element.[1]

Based on these extractions, our system determines two distributions for each relation: $P_R^{Left}(n)$ and $P_R^{Right}(n)$. The P_R^{Left} distribution represents the probability that the left argument of R is a set of size n, given that the right argument is a singleton set, and likewise for P_R^{Right}. We determine the distributions from the extractions by maximum likelihood estimation:

$$P_R^{Left}(n) = \frac{|\{F \mid F_r = R, F_{n1} = n, F_{n2} = 1\}|}{|\{F \mid F_r = R, F_{n2} = 1\}|}$$

$$P_R^{Right}(n) = \frac{|\{F \mid F_r = R, F_{n2} = n, F_{n1} = 1\}|}{|\{F \mid F_r = R, F_{n1} = 1\}|}$$

For example, for the relation `is the father of`, we might see the fragment "he is the father of two children" far more often than "he is the father of twenty children." $P^{Right}_{\texttt{is the father of}}$ would therefore have a relatively low probability for $n = 20$. As one would expect, the relation `visited` appears more often with "twenty," and the relation `married` never does. Their P^{Right} distributions are comparatively higher and lower, respectively than the one for `is the father of` at $n = 20$.

In practice, we create histograms of the extracted counts for both our E and F extractions, and our probability distributions are really distributions over the buckets in these histograms, rather than over all possible set sizes. To help combat sparse counts for large numeric values, we use buckets of exponentially increasing width for larger numeric values. Thus between $n = 0$ and 10, buckets have size 1, between 10 and 100 they have size 10, and so on.

We also create distributions in the same way for relations together with their extracted argument classes. Since counts for these extractions tend to be much more sparse, we interpolate these distributions with the distribution for just the relation, and with the distribution for the relation and just one class. We use equal weights for all interpolated distributions.

5 A Probabilistic Model for Quantifier Scope Disambiguation

QSD requires reasoning about different possible states of the world. This involves logical reasoning, since the direct and indirect readings differ in the number of objects that exist in models satisfying each reading, and the number of relationships between those objects. QSD also involves probabilistic reasoning, since none of the extracted knowledge is certain. We leverage recent work on Markov Logic Networks (MLNs) (Richardson and Domingos, 2006) to incorporate both types of reasoning into our technique for QSD. We next briefly review MLNs, before describing our model and

[1] An example of an exception to this rule from our data set is the sentence "It is worn by millions of women." Here, "it" refers to a class of items such as a brand, and thus may refer to a different item for each of the "millions of women."

methods for training it.

5.1 Markov Logic Networks

Syntactically, an MLN consists of a set of first-order logical formulas $\mathbf{F}$ and a real-valued weight w_F for each $F \in \mathbf{F}$. Semantically, an MLN defines a probability distribution over possible *groundings* of the logical formulas. That is, if U denotes the set of all objects in the universe, and $\mathbf{G}$ denotes the set of all possible ways to ground every $F \in \mathbf{F}$ (*i.e.*, substitute an element from U for every variable in F), then an MLN defines a distribution over truth assignments to the grounded formulas $G \in \mathbf{G}$. Let $\mathcal{I}$ denote the set of all possible interpretations of $\mathbf{G}$ — that is, each $I \in \mathcal{I}$ assigns `true` or `false` to every $G \in \mathbf{G}$. The probability of a particular interpretation I according to the MLN is given by :

$$P(I) = \frac{1}{Z} \exp\left(\sum_{F \in \mathbf{F}} w_F \cdot n(F, I)\right)$$

$$Z = \sum_{I \in \mathcal{I}} \exp\left(\sum_{F \in \mathbf{F}} w_F \cdot n(F, I)\right)$$

where $n(F, I)$ gives the number of groundings of F that are true in interpretation I.

The equation above provides an expression for $P(I)$ when U, or at least the size of U, is known and fixed. When we are interpreting expressions like "every city" or "every doctor", however, we require extracted knowledge to inform the system of the correct number of "city" or "doctor" objects. Since our extractions are uncertain, they provide a distribution $P(|U| = n)$ for the size of a class. Using $P(|U|)$, we can still calculate $P(I)$, even without knowing the exact size of U:

$$P(I) = \sum_n P(|U| = n) P(I \mid |U| = n)$$

5.2 MLN Classifier for QSD

Let Q be a QSD problem, consisting of a relation Q_r, a class for the first argument of the relation Q_{c1}, a class for the second argument Q_{c2}, and quantifiiers Q_{q1}, Q_{q2} for each argument. We construct an MLN model for Q using the following logical formulas:

1) *Clustering:* We allow members of each class to belong to clusters denoted by γ, but each element can belong to no more than one cluster. This is represented by the following formula, which has infinite weight.

$$\forall_{x \in Q_{c1} \cup Q_{c2}, \gamma, \gamma'} x \in \gamma \wedge x \in \gamma' \Rightarrow \gamma = \gamma'$$

2) *Relation between clusters:* Every cluster of class 1 elements must participate in the relation Q_r with exactly one cluster of class 2 elements, and *vice versa*. We represent this participation in Q_r with a series of logical relations $R_{m,n}$, each of which indicates that a cluster of size m is participating in Q_r with a cluster of size n. We use a set of formulas for each setting of m and n, each having infinite weight.

$$\forall_{\gamma \subset Q_{c1}} \exists!_{\gamma' \subset Q_{c2}, m, n} \; R_{m,n}(\gamma, \gamma')$$
$$\forall_{\gamma' \subset Q_{c2}} \exists!_{\gamma \subset Q_{c1}, m, n} \; R_{m,n}(\gamma, \gamma')$$
$$\forall_{\gamma, \gamma'} \; R_{m,n}(\gamma, \gamma') \Rightarrow |\gamma| = m \wedge |\gamma'| = n$$

3) *Prefer relations between clusters of the appropriate size:* We include a set of formulas with finite weight that express the preference for a particular relation to have arguments of a certain size. There is a separate formula for each setting of m and n, with a separate weight $w_{m,n}$ for each.

$$\forall_{\gamma, \gamma'} R_{m,n}(\gamma, \gamma')$$

This formula does most of the work of our classifier. For a given relation, such as the `lives in(Person, City)` relation, we can set the weights $w_{m,n}$ so that the model prefers worlds where each person lives in just one place. For instance, we can set the weight $w_{1,1}$ relatively high, so that the model is more likely to make clusters of size 1, which then participate in the $R_{1,1}$ relation.

We describe how we choose the $w_{m,n}$ weights below, but first we explain how to incorporate the quantifiers Q_{q1} and Q_{q2} into the model. Unfortunately, every natural language quantifier has different semantics (Barwise and Cooper, 1981), and thus they affect our model in different ways. Here, we restrict our attention to the two common quantifiers "a" and "every", but note that the MLN framework is a powerful tool for incorporating the logical semantics and statistical preferences of other quantifiers.

For the quantifier "a", we require that the relation have no argument clusters with size more than 1 for that class. Thus if Q_{q1} = "a", we restrict $R_{m,n}$ to $R_{1,n}$, and *vice versa* if Q_{q2} = "a." Furthermore, we require that at least one element of the class belong to a cluster: $\exists_{x,\gamma} x \in \gamma$ has infinite weight. For "every," we require that every element of the class that "every" modifies to be part

of some cluster. To effect this change, we simply put an infinite weight on the formula $\forall_x \exists_\gamma x \in \gamma$.

Our MLN model is general in the sense that for any QSD problem Q, it can determine probabilities for any possible world corresponding to a reading of Q. For our purposes, we are primarily interested in the direct and indirect readings of any Q involving "a" and "every." To predict the correct reading for a given Q, we simply check to see which has the higher probability according to our MLN model.

5.3 Parameter Estimation

Our MLN model for QSD requires settings for the $w_{m,n}$ parameters for each QSD problem Q. The standard approach to this problem would be to estimate these parameters from labeled training data. We reject the standard supervised framework, however, because each distinct relation Q_r requires different settings of the parameters, and therefore a standard supervised approach would require manually labeled training data for every relation Q_r.

A second approach that is made possible by our extraction technique is to set the parameters using the extracted distributions. We tried this approach by setting $w_{1,n} = \log P_{Q_r}^{Right}(n)$ and $w_{m,1} = \log P_{Q_r}^{Left}(m)$; since we only consider sentences containing the quantifier "a", one of m and n will always be 1. Unfortunately, in our experiments we found that this setting for the parameters often gave far too little weight for large values of m and n, and as a consequence, the classifier would systematically judge one reading to be more likely than another.

To counteract this problem, we take a hybrid approach to parameter estimation, informed by both labeled training data and the extracted distributions. Crucially, our approach, which we call ZIPF FLATTENING, has only two parameters that need to be trained using a supervised approach, and these parameters do not depend on the relation R. Thus, the approach minimizes the amount of training data we need to a practical level.

ZIPF FLATTENING works by correcting the P_R distributions to give higher weight to larger values of m and n. First, we estimate a Zipf distribution from the raw extracted counts for each argument of relation R. To fit a Zipf curve, we use least-squares linear regression on the log-log plot of the extracted counts to find parameters z_R and c_R such that

$$\begin{aligned} \log(count) &= z_R \cdot \log(argSize) + c_R \\ \Rightarrow count &= e^{c_R} \cdot argSize^{z_R} \end{aligned}$$

We can perform this part automatically, using only the extraction data and no manually labeled training data, for every relation. However, the fitted Zipf distribution needs to be corrected for the systematic bias in the extracted counts. To do this, we introduce two parameters, α_1 and α_2, that we use to scale back the sharp falloff in the Zipf distribution. Our *flattened* distribution has the form:

$$count = e^{\alpha_1 c_R} \cdot argSize^{\alpha_2 z_R}$$

When α_2 is less than 1, the resulting curve has a less steep slope, and greater weight is placed on the large values of m and n, as desired. Our last step is to interpolate the P_R^{Right} and P_R^{Left} distributions with the flattened Zipf distribution to come up with corrected distributions for the right and left argument sizes of R. We use equal weights on the two distributions to interpolate. Note that if the original counts from the extraction system include counts for only one argument size, then it is impossible to estimate a Zipf distribution, and we simply fall back on the extracted distribution. We do not include counts for an argument size of zero in this process.

To estimate the parameters α_i, we collect a training set of QSD problems Q, labeled with the correct reading for each (direct or indirect), and run the extractor for the relations Q_r appearing in the training set. We then perform a gradient descent search to find optimal settings for the α_i on the training data.

6 Experiments

We report on two sets of experiments. The first tests our extraction technique on its own, and the second tests the accuracy of our complete QSD system, including the extraction mechanisms and the prediction model, on a quantifier scope disambiguation task.

6.1 Function Detection Experiment

Function detection is an important task in its own right, and has been used in several previous applications (Ritter et al., 2008; Yates and Etzioni, 2009; Popescu, 2007). To turn our extraction system into a classifier for functions vs. non-functions, we simply checked whether there were

	Num	Precision	Recall	F1
Functions	54	.79	.76	.77
Non-functions	74	.83	.85	.84

Table 3: **Precision and recall for detecting functions using the numeric extraction technique.**

any extractions for R with $F_{n2} > 1$. If so, we predicted that the R was nonfunctional, and otherwise we predicted it was functional.

We used the Web1Tgram Corpus of n-grams provided by Google, Inc to extract classes, relations, and counts. This corpus contains counts for 2- through 5-grams that appear on the Web pages indexed by Google. Counts are included in this data set for all n-grams that appeared at least 40 times in their text. We ran our extraction techniques on the 3-, 4- and 5-grams. To create a test set, we sampled a set of 200 relations from our extractions, removed any relations that consisted of punctuations, stopwords, or other non-relational items. We then manually labeled the remainder as functions or non-functions.

Table 3 shows our results. A baseline system that simply predicts the majority class (non-functions) on this data set would achieve an accuracy of 56%, well below the 81% accuracy of our classifier. Many of the relations in our test set, like `built(Person, House)` and `is riding(Person,Animal)`, do not ordinarily have named-entity extractions for both arguments, and would therefore not be amenable to previous function detection approaches.

Some of our technique's errors highlight interesting difficulties with function detection. For instance, while we labeled the `is capital of` relation as a function, our technique predicted that it was not. It turns out that the country of Bolivia has two capitals, and the South Asian region of Jammu and Kashmir also has two capitals. Both of these facts are prominent enough on the Web to cause our system to detect a small probability for $P^{Right}_{capital\ of}(2)$. Thus any label for this relation is somewhat unsatisfying: it is almost entirely functional, but not strictly so. By generalizing the problem to one of determining a distribution for the size of the argument, we can handle these border cases in a useful way for QSD, as discussed below.

6.2 Preliminary QSD Experiments

We test our complete QSD system on two important tasks. In the first, the system is presented with a series of QSD problems Q in which the first quantifier Q_{q1} is always "a," and the second (Q_{q2}) is always "every." Each example is manually labeled to indicate whether a direct or indirect reading of the sentence is preferred, and the system is charged with predicting the preferred reading. In the second task, each Q has "every" as the first quantifier, and "a" as the second quantifier. Since indirect readings are very rarely preferred for active-voice sentences of this form,we charge the system with making a different type of prediction: determine whether the indirect reading is plausible or not. The system assumes that every sentence has a plausible direct reading, but by determining whether the indirect reading is plausible, it can determine whether the sentence is ambiguous between the two readings.

We created data sets for these tasks by sampling our 5grams for examples containing the relations in our function experiment. From this set, we selected phrases that involved named classes for the arguments to the relation. When a class was missing, we either manually supplied one, or discarded the example. We then constructed two examples from each combination of relation and argument classes: one example in which the first argument is constrained by the quantifier "a" and the second by "every," and a second example in which the quantifiers are reversed. Finally, we manually labeled every example with a preference for direct or indirect reading (in the case of "a/every" examples) or with a plausibility judgment for the indirect reading (in the case of "every/a" examples). Our final test sets included 46 labeled examples for each task. Further experiments involving multiple annotators, as in the experiments of Kurtzman and MacDonald (1993), are of course desirable, but note that even their experiments included just 32 labeled examples.

Table 4 shows our results for the first QSD task, and Table 5 shows our results for the second one. In each case, we compare our supervised Corrected MLN model against an Uncorrected MLN model that uses no supervised data, and simply takes its weights straight from our extracted distributions. The supervised model uses a training corpus of 10 manually labeled examples for each task, five from each class. We also compare against a majority class baseline. Note that the Corrected

System	Acc.	Direct P	Direct R	Indirect P	Indirect R
All-Direct BL	.53	.53	1.0	0.0	0.0
Uncorrected MLN	.58	.78	.30	.53	.90
Corrected MLN	**.74**	.77	.74	.71	.75

Table 4: **Our trained MLN outperforms two other systems at predicting whether sentences of the form "A/some <class 1> <relation> every <class 2>" should have direct or indirect readings.** We measure accuracy over the whole dataset, as well as precision and recall for the two subsets labeled with direct and indirect readings, respectively.

System	Acc.	Plausible P	Plausible R	Implaus. P	Implaus. R
All-Plausible BL	.67	.67	1.0	0.0	0.0
Uncorrected MLN	.49	.89	.28	.38	.93
Corrected MLN	**.72**	.76	.86	.60	.43

Table 5: **Our trained MLN outperforms two other systems at predicting whether sentences of the form "Every <class 1> <relation> a/some <class 2>" have a plausible indirect reading or not.** We measure accuracy over the whole dataset, as well as precision and recall for the two subsets labeled with plausible and implausible indirect readings.

MLN model has balanced recall numbers for the two classes in both of our tasks, compared with the Uncorrected MLN. This indicates that our ZIPF FLATTENING technique is accurately learning better weights to remove the systematic bias in the Uncorrected MLN.

Our results demonstrate the utility of our extracted distributions for these difficult tasks. Although the extracted data prevents us from determining that `is capital of` should be classified as a function, since almost all of the probability mass in P^{Right} is still on $n \in \{0, 1\}$. Thus, the probability for the direct reading of a sentence like "Some city is the capital of every country" is still very low. Likewise, even though our system (correctly) determines that the relation `is a parent of` is non-functional, it does not therefore group it with other non-functional relations like `visited`. The distribution $P^{Right}_{\texttt{is parent of}}(n)$ is skewed to much smaller numbers for n than is the distribution for `visited`, and thus the indirect reading for "A person is the parent of every child" is much more likely than the indirect reading of "A person visited every country."

The biggest hurdle for better performance is noise in our extraction technique. Polysemous relations sometimes have large counts for large argument sizes in one sense, but not another. Using argument classes to disambiguate relations can help, but extractions for relations in combination with argument classes are much more sparse. Improved extraction techniques could directly impact performance on the QSD task.

7 Conclusion and Future Work

We have demonstrated targeted methods for extracting world knowledge that is necessary for making quantifier scope disambiguation decisions. We have also demonstrated a novel, minimally-supervised, statistical relational model in the Markov Logic Network framework for making QSD decisions based on extracted pragmatics.

While our preliminary results for QSD are promising, there are clearly many areas for improvement. We will need to handle more kinds of quantifiers in our MLN model. Our current system is biased towards using purely pragmatic knowledge, but a complete system should also integrate syntactic and lexical constraints and preferences. Also, discourses can introduce knowledge that directly affects QSD problems, such as constraints on the size of a particular set that is discussed in the discourse. Integrating our technique for QSD with discourse processing is a major challenge that we hope to address.

References

Hiyan Alshawi. 1990. Resolving quasi logical forms. *Computational Linguistics*, 16(3):133–144.

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional information extraction. In *Proceedings of the ACL*.

J. Barwise and R. Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):150–219.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3:281–332.

D. Davidov and A. Rappaport. 2008. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated SAT analogy questions. In *Proceedings of the ACL*.

Doug Downey, Oren Etzioni, and Stephen Soderland. 2005. A Probabilistic Model of Redundancy in Information Extraction. In *IJCAI*.

Jerry R. Hobbs and Stuart M. Shieber. 1987. An algorithm for generating quantifier scopings. *Computational Linguistics*, 13(1-2):47–63.

Sven Hurum. 1988. Handling scope ambiguities in English. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 58–65.

Howard S. Kurtzman and Maryellen C. MacDonald. 1993. Resolution of quantifier scope ambiguities. *Cognition*, 48:243–279.

Longin Latecki. 1992. Connection relations and quantifier scope. In *Proceedings of the ACL*.

Richard Montague. 1973. The proper treatment of quantification in ordinary English. In Jaakko Hintikka, Julius Moravcsik, and Patrick Suppes, editors, *Approaches to Natural Languages*, pages 221–242. Reidel, Dordrecht.

Douglas B. Moran. 1988. Quantifier scoping in the SRI core language engine. In *Proceedings of the 26th Annual Meeting of the Assoc. for Comp. Linguistics*, pages 33–40.

Jong C. Park. 1988. Quantifier scope and constituency. In *Proceedings of the 26th Annual Meeting of the Assoc. for Comp. Linguistics*, pages 33–40.

Massimo Poesio. 1993a. Assigning a semantic scope to operators. In *Proceedings of the ACL*.

Massimo Poesio. 1993b. Assigning a semantic scope to operators. In *Proceedings of the Second Conference on Situation Theory and Its Applications*.

Ana-Maria Popescu. 2007. *Information Extraction from Unstructured Web Text*. Ph.D. thesis, University of Washington.

Uwe Reyle. 1995. On reasoning with ambiguities. In *Proceedings of the EACL*, pages 1–8.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62:107–136.

Alan Ritter, Doug Downey, Stephen Soderland, and Oren Etzioni. 2008. It's a contradiction — No, it's not: A case study using functional relations. In *Empirical Methods in Natural Language Processing*.

Walid S. Saba and Jean-Pierre Corriveau. 1997. A pragmatic treatment of quantification in natural language. In *Proceedings of the National Conference on Artificial Intelligence*.

Walid S. Saba and Jean-Pierre Corriveau. 2001. Plausible reasoning and the resolution of quantifier scope ambiguities. *Studia Logica*, 67:271–289.

Stefan Schoenmackers, Oren Etzioni, and Dan Weld. 2008. Scaling textual inference to the web. In *Proceedings of EMNLP*.

Kurt Van Lehn. 1978. Determining the scope of English quantifiers. Technical Report AI-TR-483, AI Lab, MIT.

Alexander Yates and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research (JAIR)*, 34:255–296, March.

Alexander Yates, Stefan Schoenmackers, and Oren Etzioni. 2006. Detecting parser errors using web-based semantic filters. In *Proceedings of EMNLP*.

Chinese Semantic Role Labeling with Shallow Parsing

Weiwei Sun and **Zhifang Sui** and **Meng Wang** and **Xin Wang**
Institute of Computational Linguistics
Peking University
Key Laboratory of Computational Linguistics
Ministry of Education, China
weiwsun@gmail.com;{szf,wm}@pku.edu.cn;xinwang.cpku@gmail.com;

Abstract

Most existing systems for Chinese Semantic Role Labeling (SRL) make use of full syntactic parses. In this paper, we evaluate SRL methods that take partial parses as inputs. We first extend the study on Chinese shallow parsing presented in (Chen et al., 2006) by raising a set of additional features. On the basis of our shallow parser, we implement SRL systems which cast SRL as the classification of syntactic chunks with IOB2 representation for semantic roles (i.e. semantic chunks). Two labeling strategies are presented: 1) directly tagging semantic chunks in one-stage, and 2) identifying argument boundaries as a chunking task and labeling their semantic types as a classification task. For both methods, we present encouraging results, achieving significant improvements over the best reported SRL performance in the literature. Additionally, we put forward a rule-based algorithm to automatically acquire Chinese verb formation, which is empirically shown to enhance SRL.

1 Introduction

In the last few years, there has been an increasing interest in Semantic Role Labeling (SRL) on several languages, which consists of recognizing arguments involved by predicates of a given sentence and labeling their semantic types. Nearly all previous Chinese SRL research took full syntactic parsing as a necessary pre-processing step, such as (Sun and Jurafsky, 2004; Xue, 2008; Ding and Chang, 2008). Many features are extracted to encode the complex syntactic information. In English SRL research, there have been some attempts at relaxing the necessity of using full syntactic parses; better understanding of SRL with shallow parsing is achieved by CoNLL-2004 shared task (Carreras and Màrquez, 2004). However, it is still unknown how these methods perform on other languages, such as Chinese.

To date, the best SRL performance reported on the Chinese Proposition Bank (CPB) corresponds to a F-measure is 92.0, when using the hand-crafted parse trees from Chinese Penn Treebank (CTB). This performance drops to 71.9 when a real parser is used instead[1] (Xue, 2008). Comparatively, the best English SRL results reported drops from 91.2 (Pradhan et al., 2008) to 80.56 (Surdeanu et al., 2007). These results suggest that as still in its infancy stage, Chinese full parsing acts as a central bottleneck that severely limits our ability to solve Chinese SRL. On the contrary, Chinese shallow parsing has gained a promising result (Chen et al., 2006); hence it is an alternative choice for Chinese SRL.

This paper addresses the Chinese SRL problem on the basis of shallow syntactic information at the level of phrase chunks. We first extend the study on Chinese chunking presented in (Chen et al., 2006) by raising a set of additional features. The new set of features yield improvement over the strong chunking system described in (Chen et al., 2006). On the basis of our shallow parser, we implement lightweight systems which solve SRL as a sequence labeling problem. This is accomplished by casting SRL as the classification of syntactic chunks (e.g. NP-chunk) into one of semantic labels with IOB2 representation (**?**). With respect to the labeling strategy, we distinguish two different approaches. The first one directly recognizes semantic roles by an IOB-type sequence tagging. The second approach divides the problem into two independent subtasks: 1) Argument Identification (AI) and 2) Semantic Role Classification (SRC).

[1] This F-measure is evaluated on the basis of hand-crafted word segmentation and POS tagging.

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1475–1483,
Singapore, 6-7 August 2009.

A Chinese word consists of one or more characters, and each character, in most cases, is a morpheme. The problem of how the words are constructed from morphemes, known as word formation, is very important for a majority of Chinese language processing tasks. To capture Chinese verb formation information, we introduce a rule-based algorithm with a number of heuristics. Experimental results indicate that word formation features can help both shallow parsing and SRL.

We present encouraging SRL results on CPB[2]. The best F-measure performance (74.12) with gold segmentation and POS tagging can be achieved by the first method. This result yield significant improvement over the best reported SRL performance (71.9) in the literature (Xue, 2008). The best recall performance (71.50) can be achieved by the second method. This result is also much higher than the best reported recall (65.6) in (Xue, 2008).

2 Related Work

Previous work on Chinese SRL mainly focused on how to implement SRL methods which are successful on English, such as (Sun and Jurafsky, 2004; Xue and Palmer, 2005; Xue, 2008; Ding and Chang, 2008). Sun and Jurafsky (2004) did the preliminary work on Chinese SRL without any large semantically annotated corpus of Chinese. Their experiments were evaluated only on ten specified verbs with a small collection of Chinese sentences. This work made the first attempt on Chinese SRL and produced promising results. After the CPB was built, (Xue and Palmer, 2005) and (Xue, 2008) have produced more complete and systematic research on Chinese SRL. Ding and Chang (2008) divided SRC into two sub-tasks in sequence. Under the hierarchical architecture, each argument should first be determined whether it is a core argument or an adjunct, and then be classified into fine-grained categories. Chen et al. (2008) introduced an application of transductive SVM in Chinese SRL. Because their experiments took hand-crafted syntactic trees as input, how transductive SVMs perform in Chinese SRL in realistic situations is still unknown.

Most existing systems for automatic Chinese SRL make use of a full syntactic parse of the sentence in order to define argument boundaries and to extract relevant information for training classifiers to disambiguate between role labels. On the contrary, in English SRL research, there have been some attempts at relaxing the necessity of using syntactic information derived from full parse trees. For example, Hacioglu and Ward (2003) considered SRL as a chunking task; Pradhan et al. (2005) introduced a new procedure to incorporate SRL results predicted respectively on full and shallow syntactic parses. Previous work on English suggests that even good labeling performance has been achieved by full parse based SRL systems, partial parse based SRL systems can still enhance their performance. Though better understanding of SRL with shallow parsing on English is achieved by CoNLL-2004 shared task (Carreras and Màrquez, 2004), little is known about how these SRL methods perform on Chinese.

3 Chinese Shallow Parsing

There have been some research on Chinese shallow parsing, and a variety of chunk definitions have been proposed. However, most of these studies did not provide sufficient detail. In our system, we use chunk definition presented in (Chen et al., 2006), which provided a chunk extraction tool. The tool to extract chunks from CTB was developed by modifying the English tool used in CoNLL-2000 shared task, Chunklink[3], and is publicly available at http://www.nlplab.cn/chenwl/chunking.html. The definition of syntactic chunks is illustrated in Line *CH* in Figure 1. For example, "保险公司/the insurance company", consisting of two nouns, is a noun phrase.

With IOB2 representation (Ramshaw and Marcus, 1995), the problem of Chinese chunking can be regarded as a sequence labeling task. In this paper, we first implement the chunking method described in (Chen et al., 2006) as a strong baseline. To conveniently illustrate, we denote a word in focus with a fixed window $w_{-2}w_{-1}ww_{+1}w_{+2}$, where w is current token. The baseline features includes:

- Uni-gram word/POS tag feature: w_{-2}, w_{-1}, w, w_{+1}, w_{+2};
- Bi-gram word/POS tag feature: $w_{-2}\_w_{-1}$, w_{-1}_w, w_w_{+1}, $w_{+1}\_w_{+2}$;

[2]Our system is available at http://code.google.com/p/csrler/

[3]http://ilk.uvt.nl/team/sabine/chunklink/chunklink_2-2-2000_for_conll.pl

WORD:	截止	目前	保险	公司	已	为	三峡	工程	提供	保险	服务
POS:	[P]	[NT]	[NN	NN]	[AD]	[P]	[NR]	[NN]	[VP]	[NN	NN]
CH:	[PP	NP]	[NP]		[ADVP]	[PP	NP	NP]	[VP]	[NP]	
M1:	B-A*	I-A*[4]	B-A0		B-AM-ADV	B-A2	I-A2	I-A2	B-V	B-A1	
M2-AI:	B-A	I-A	B-A		B-A	B-A	I-A	I-A	B-V	B-A	
M2-SRC:	AM-TMP		A0		AM-ADV		A2		Rel	A1	

Until now, the insurance company has provided insurance services for the Sanxia Project.

Figure 1: An example from Chinese PropBank.

That means 18 features are used to represent a given token. For instance, the bi-gram Word features at 5th word position ("公司/company") in Figure 1 are "，_保险", "保险_公司", "公司_已", "已_为".

To improve shallow parsing, we raised an additional set of features. We will discuss these features in section 5.

4 SRL with Shallow Parsing

The CPB is a project to add predicate-argument relations to the syntactic trees of the CTB. Similar to English PropBank, the semantic arguments of a predicate are labeled with a contiguous sequence of integers, in the form of AN (i.e. Arg$N$); the adjuncts are annotated as such with the label AM (i.e. ArgM) followed by a secondary tag that represents the semantic classification of the adjunct. The assignment of argument labels is illustrated in Figure 1, where the predicate is the verb "提供/provide". For example, the noun phrase "保险公司/the insurance company" is labeled as *A0*, meaning that it is the *proto-Agent* of 提供; the preposition phrase "截止目前/until now" is labeled as *AM-TMP*, indicating a temporal component.

4.1 System Architecture

SRL is a complex task which has to be decomposed into a number of simpler decisions and tagging schemes in order to be addressed by learning techniques. Regarding the labeling strategy, we can distinguish at least two different strategies. The first one consists of performing role identification directly as IOB-type sequence tagging. The second approach consists of dividing the problem into two independent subtasks.

[4]The semantic chunk labels here are *B-AM-TMP* and *I-AM-TMP*. Limited to the document length, we cannot put all detailed chunk labels in one line in Figure 1.

4.1.1 One-stage Strategy

In the one-stage strategy, on the basis of syntactic chunks, we define semantic chunks which do not overlap nor embed using IOB2 representation. Syntactic chunks outside a chunk receive the tag O. For syntactic chunks forming a chunk of type *A**, the first chunk receives the *B-A** tag (Begin), and the remaining ones receive the tag *I-A** (Inside). Then a SRL system can work directly by using sequence tagging techinique. Since the semantic annotation in the PropBank corpus does not have any embedded structure, there is no loss of information in this representation. The line *M1* in Figure 1 illustrates this semantic chunk definition.

4.1.2 Two-stage Strategy

In the two-stage architecture, we divide Chinese SRL into two subtasks: 1) semantic chunking for AI, in which the argument boundaries are predicted, and 2) classification for SRC, in which the already recognized arguments are assigned role labels. In the first stage, we define semantic chunks *B-A* which means begin of an argument and *I-A* which means inside of an argument. In the second stage, we solve SRC problem as a multi-class classification. The lines *M2-AI* and *M2-SRC* in Figure 1 illustrate this two-stage architecture. For example, the noun phrase "保险公司/the insurance company" is *proto-Agent*, and thus should be labeled as *B-A* in the AI chunking phase, and then be tagged as *A0*. The phrase "为三峡工程/for the Sanxia Project" consists of three chunks, which should be labeled as *B-A*, *I-A*, and *I-A* respectively in the AI chunking phase, then these three chunks as a whole argument should be recognized as *A2*.

4.1.3 Chunk-by-Chunk

There is also another semantic chunk definition, where the basic components of a semantic chunk are words rather than syntactic chunks. A good election for this problem is chunk-by-chunk pro-

cessing instead of word-by-word. The motivation is twofold: 1) phrase boundaries are almost always consistent with argument boundaries; 2) chunk-by-chunk processing is computationally less expensive and allows systems to explore a relatively larger context. This paper performs a chunk-by-chunk processing, but admitting a processing by words within the target verb chunks.

4.2 Features

Most of the feature templates are "standard", which have been used in previous SRL research. We give a brief description of "standard" features, but explain our new features in detail.[5]

4.2.1 Features for Semantic Chunking

In the semantic chunking tasks, i.e. the one-stage method and the first step in the two-stage method, we use the same set of features. The features are extracted from three types of elements: syntactic chunks, target verbs, links between chunks and target verbs. They are formed making use of words, POS tags and chunks of the sentence. Xue (2008) put forward a rough verb classification where verb classes are automatically derived from the frame files, which are verb lexicon for the CPB annotation. This kind of verb class information has been shown very useful for Chinese SRL. Our system also includes this feature. In our experiments, we represent a verb in two dimensions: 1) number of arguments, and 2) number of framesets. For example, a verb may belong to the class "C1C2," which means that this verb has two framesets, with the first frameset having one argument and the second having two arguments.

To conveniently illustrate, we denote a token chunk with a fixed context $w_{i-1}[_{c_k} w_i ... w_h ... w_j]w_{j+1}$, where w_h is the head word of this chunk c_k. The complete list of features is listed here.

Extraction on Syntactic Chunks

Chunk type: c_k.

Length: the number of words in a chunk.

Head word/POS tag. The rules described in (Sun and Jurafsky, 2004) are used to extract head word.

IOB chunk tag of head word: chunk tag of head word with IOB2 representation (e.g. B-NP, I-NP).

Chunk words/POS tags context. Chunk context includes one word before and one word after: w_{i-1} and w_{j+1}.

POS tag chain: sequential containers of each word's POS tag: $w_{i\_...\_}w_j$. For example, this feature for "保险服务" is "NN_NN".

Position: the position of the phrase with respect to the predicate. It has three values as *before*, *after* and *here*.

Extraction on Target Verbs Given a target verb w_v and its context, we extract the following features.

Predicate, its POS tag, and its verb class.

Predicate IOB chunk tag context: the chain of IOB2 chunk tags centered at the predicate within a window of size -2/+2.

Predicate POS tag context: the POS tags of the words that immediately precede and follow the predicate.

Number of predicates: the number of predicates in the sentence.

Extraction on Links To capture syntactic properties of links between the chunks and the verbs, we use the following features.

Path: a flat path is defined as a chain of base phrases between the token and the predicate. At both ends, the chain is terminated with the POS tags of the predicate and the headword of the token.

Distance: we have two notions of distance. The first is the distance of the token from the predicate as a number of base phrases, and the second is the same distance as the number of VP chunks.

Combining Features We also combine above features as some new features.

Conjunctions of position and head word, target verb, and verb class, including: $position\_w_h$, $position\_w_v$, $position\_w_h\_w_v$, $position\_class$, and $position\_w_h\_class$.

Conjunctions of position and POS tag of head word, target verb, and verb class, including: $position\_w_h\_w_v$, $position\_w_h$, and $position\_w_h\_class$.

4.2.2 Features for SRC

In the SRC stage of the two-stage method, different from previous work, our system only uses word-based features, i.e. features extracted from words and POS tags, to represent a given argument. Experiments show that a good semantic

[5]The source code of our system also provides lots of comments for implementation of all features.

role classifier can be trained by using only word-based features. To gather all argument position information predicted in AI stage, we design a coarse frame feature, which is a sequential collection of arguments. So far, we do not know the detailed semantic type of each argument, and we use *XP* as each item in the frame. To distinguish the argument in focus, we use a special symbol to indicate the corresponding frame item. For instance, the Frame feature for argument 保险服务 is *XP+XP+XP+XP+V+!XP*, where *!XP* means that it is the argument in focus.

Denote 1) a given argument $w_{i-2}w_{i-1}[w_iw_{i+1}...w_{j-1}w_j]w_{j+1}w_{j+2}$, and 2) a given predicate w_v. The features for SRC are listed as follows.

Words/POS tags context of arguments: the contents and POS tags of the following words: w_i, w_{i-1}, w_{i-2}, w_{i+1}, w_{i+2}, w_j, w_{j+1}, w_{j-1}, w_{j-2}, w_{j+1}, w_{j+2}; the POS tags of the following words: w_{i+1}, w_{i+2}, w_{j+1}, w_{j+2}.

Token Position.

Predicate, its POS, and its verb class.

Coarse Frame.

Combining features: conjunctions of boundary words, including $w_{i-1}\_w_{j+1}$ and $w_{i-2}\_w_{j+2}$; conjunction of POS tags of boundary words, including $w_{i-1}\_w_{j+1}$ and $w_{i-2}\_w_{j+2}$; conjunction of token position, boundary words, and predicate word, including $position\_w_i\_w_j$, $w_i\_w_j\_w_v$; $position\_w_i\_w_j\_w_v$; conjunction of token position, boundary words' POS tags, and predicate word, also including $position\_w_i\_w_j$, $w_i\_w_j\_w_v$; $position\_w_i\_w_j\_w_v$; conjunction of predicate and frame; conjunction of target verb class and frame; conjunction of boundary words' POS tags, and predicate word.

5 Automatic Chinese Verb Formation Analyzing

5.1 Introduction to Chinese Word Formation

Chinese words consist of one or more characters, and each character, in most cases, is a morpheme which is the smallest meaningful unit of the language. According to the number of morphemes, the words can be grouped into two sets, simple words (consisting of one morpheme) and compound words (consisting of two morphemes or more). There are 9 kinds of word formation in Chinese compound words, and table 1 shows the detail with examples. Note that, *attributive-head* and *complementarity* are not for Chinese verbs.

Types	Examples
reduplication	看看(look) 想想(think)
affixation	激化(intensify) 觉着(feel)
subject-verb	耳闻(hear) 口述(dictate)
verb-object	戒烟(quit smoking) 理发(haircut)
verb-complement	通知(inform) 栽培(plant)
verb-result	超出(exceed) 煮沸(boil)
adverbial-head	隐居(retreat) 误用(misuse)
coordinate	爱惜(cherish) 追逐(chase)
attributive-head*	谣言(rumor) 医院(hospital)
complementarity*	纸张(paper) 马匹(horse)

Table 1: Example Words with Formation

The internal structure of a word constraints its external grammatical behavior, and the formation of a verb can provide very important information for Chinese SRL. Take "超出/exceed" as an example, the two characters are both verbal morphemes, and the character "超" means "pass" and the character "出" with the meaning of "over" shows the complement of the action of "超". In this word, "出" is usually collocated with an object, and hence a *Patient* role should comes after the verb "超出". Note that, the verb "超", however, is unlikely to have an object. Take "理发/haircut" as another example, the first character "理" is a verbal morpheme with the meaning of "cut" and the second character "发" is a nominal morpheme with the meaning of "hair". In this word, "发" acts as the object of "理", and the word "理发" is unlikely to have an *Patient* any more in the sentential context.

5.2 Verb Formation Analyzing Method

To automatically analyze verb formation, we introduce a rule-based algorithm. Pseudo code in Algorithm 1 illustrates our algorithm. This algorithm takes three string (one or more Chinese characters) sets as *lexicon* knowledge:

- adverbial suffix set $\mathcal{A}$: strings in $\mathcal{A}$ are usually realized as the modifier in a *adverbial-head* type word, e.g. 不/not, 别/not, 总/always, 并/both, 共/all.

- object head set $\mathcal{O}$: strings in $\mathcal{O}$ are usually realized as the head in a *verb-object* type word, e.g. 变/change, 获/get, 谈/talk, 发/send.

Algorithm 1: Verb Formation Analyzing.

Data: adverbial suffix set $\mathcal{A}$, object head set $\mathcal{O}$, complement suffix set $\mathcal{C}$
input : word $W = c_1...c_n$ and its POS P
output: head character h, adverbial character a, complement character c, object character o

begin
  $h = c = a = o = null$;
  if $n = 4$ **and** $c_1 = c_3$ **and** $c_2 = c_4$ **then**
    return *Verb formation of* $W' = c_1c_3$;
  else if $n = 3$ **and** $c_2 = c_3$ **then**
    $h = c_1, c = c_2$;
  else if $n = 2$ **and** $c_1 = c_2$ **then**
    $h = c_1$;
  else if $n = 1$ **then**
    $h = c_1$;
  else if $c_n \in \mathcal{C}$ **and** $c_{n-1}c_n \in \mathcal{C}$ **and** P=*"VV"* **then**
    $h = c_1, c = c_n/c_{n-1}c_n$;
  else if $c_1 \in \mathcal{A}$ **then**
    $a = c_1, h = c_2...c_n$;
  else if $c_1 \in \mathcal{O}$ **and** P=*"VV"* **then**
    $h = c_1, o = c_2...c_n$;
end

- complement suffix set $\mathcal{C}$: strings in $\mathcal{C}$ are usually realized as complement in a verb-complement type word: e.g. 出/out, 入/in, 完/finish, 来/come, 不到/not.

Note that, to date there is no word formation annotation corpus, so direct evaluation of our rule-based algorithm is impossible. This paper makes task-oriented evaluation which measures improvements in SRL.

5.3 Using Word Formation Information to improve Shallow Parsing

The majority of Chinese nouns are of type *attributive-head*. This means that for most nouns the last character provides very important information indicating the head of the noun. For example, the word formations of "桃树/peach", "柳树/willow" and "黄杨树/boxtree" (three different kinds of trees), are *attributive-head* and they have the same head word "树/tree". While for verbs, the majority are of three types: *verb-object*, *coordinate* and *adverbial-head*. For example, words "加大/enlarge", "加剧/make more drastic" and "加快/accelerate" have the same *head* "加/add". The *head* morpheme is very useful in alleviating the data sparseness in word level. However, for any given word, it is very hard to accurately find the head. In the shallow paring experiments, we use a very simple rule to get a *pseudo* head character: 1) extracting the last word for a noun, and 2) extracting the first word for a verb. The new features include:

Pattern 1: conjunction of pseudo head of w_{i-1} and POS tags of w_{i-1} and w_i.

Pattern 2: conjunction of pseudo head of w_i and POS tags of w_{i-1} and w_i.

Pattern 3: conjunction of length/POS tags of w_{i-1}, w_i, w_{i+1}.

5.4 Using Verb Formation Information to improve SRL

We use some new verb formation features to improve our SRL system. The new features are listed as follows. The first four are used in semantic chunking task, and all are used in SRC task.

First/last characters.

Word length.

Conjunction of word length and first/last character.

Conjunction of token position and first/last character.

The head string of a verb (e.g. "理" in "理发").

The adverbial string of a verb (e.g. "误" in "误用").

The complement string of a verb (e.g. "出" in "超出").

The object string of a verb (e.g. "发" in "理发").

6 Results and Discussion

6.1 Experimental Setting

6.1.1 Data

Experiments in previous work are mainly based on CPB and CTB, but the experimental data preparing procedure does not seem consistent. For example, the sum of each semantic role reported in (Ding and Chang, 2008) is extremely smaller than the corresponding occurrence statistics in original data files in CPB. In this paper, we modify CoNLL-2005 shared task software[6] to process CPB and CTB. In our experiments, we use the CPB 1.0 and CTB 5.0. The data is divided into three parts: files from chtb_081 to chtb_899 are used as training set; files from chtb_041 to

[6]http://www.lsi.upc.edu/∼srlconll/soft.html

chtb_080 as development set; files from chtb_001 to chtb_040, and chtb_900 to chtb_931 as test set. The data setting is the same as (Xue, 2008). The results were evaluated for precision, recall and F-measure numbers using the srl-eval.pl script provided by CoNLL-2005 shared task.

6.1.2 Classifier

For both syntactic and semantic chunking, we used TinySVM along with YamCha[7] (Kudo and Matsumoto, 2000; Kudo and Matsumoto, 2001). In the chunking experiments, all SVM classifiers were realized with a polynomial kernel of degree 2. Pair-wise strategy is used to solve multi-class classification problem. For the SRC experiments, we use a linear SVM classifier, along with One-Vs-All approach for multi-class classification. SVM_{lin}[8], a fast linear SVM solvers, is used for supervised learning. l_2-SVM-MFN (modified finite newton) method is used to solve the optimization problem (Keerthi and DeCoste, 2005).

6.2 Shallow Parsing Performance

	P(%)	R(%)	$F_{\beta=1}$
Baseline	93.54	93.00	93.27
Ours	93.83	93.39	93.61

Table 2: Shallow parsing performance

Table 2 summarizes the overall shallow parsing performance on test set. The first line shows the performance of baseline. Comparing the best system performance 94.13 F-measure of CoNLL 2000 shared task (Syntactic Chunking on English), we can see Chinese shallow parsing has reached a comparable result, tough the comparison of numeric performance is not very fair, because of different languages, different chunk definition, different training data sizes, etc.. The second line *Ours* shows the performance when new features are added, from which we can see the word formation based features can help shallow parsing.

Table 3 shows the detailed performance of noun phrase (NP) and verb phrase (VP), which make up most of phrase chunks in Chinese. Our new features help NP more, whereas the effect of new features for VP is not significant. That is in part because most VP chunk recognition error is caused by long dependency, where word formation features do not work. Take the sentences below for example:

	P(%)	R(%)	$F_{\beta=1}$
NP(Baseline)	90.84	90.05	90.44
NP(Ours)	91.42	90.78	91.10
VP(Baseline)	94.44	94.55	94.50
VP(Ours)	94.65	94.74	94.69

Table 3: Performance of NP-chunk and VP-chunk

1. [$_{VP}$ 因此获得胜利]。 (Therefore (we) achieve victory.)
2. [$_{ADVP}$ 因此] [$_{VP}$ 大量出现] 的是以前不曾遇到的。 (Therefore the major changes have not been met before.)

The contexts of the word "因此/therefore" in the two sentences are similar, where "因此" is followed by verbal components. In the second sentence, the word "因此/therefore" will be correctly recognized as an adverbial phrase unless classifier knows the following component is a clause. Unfortunately, word formation features cannot supply this kind of information.

6.3 SRL Performance

	P(%)	R(%)	A(%)	$F_{\beta=1}$
(Xue, 2008)	79.5	65.6	–	71.9
M1−	79.02	69.12	–	73.74
M1+	79.25	69.61	–	74.12
M2−/AI	80.34	75.11	–	77.63
M2+/AI	80.01	75.15	–	77.51
M2−/SRC	–	–	92.57	–
M2+wf/SRC	–	–	93.25	–
M2+/SRC	–	–	93.42	–
M2−AI+SRC	76.48	71.50	–	73.90

Table 4: Overall SRL performance of different methods

Table 4 lists the overall SRL performance numbers on test set using different methods mentioned earlier; these results are based on features computed from gold standard segmentation and POS tagging, but automatic recognized chunks, which is parsed by our improved shallow parsing system. For the AI and the whole SRL tasks, we report the precision (P), recall (R) and the $F_{\beta=1}$-measure scores, and for the SRC task we report the classification accuracy (A). The first line *(Xue,*

[7] http://chasen.org/∼taku/index.html.en

[8] http://people.cs.uchicago.edu/∼vikass/svmlin.html

2008) shows the SRL performance reported in (Xue, 2008). To the authors' knowledge, this result is best SRL performance in the literature. Line 2 and 3 shows the performance of the one-stage systems: 1) Line *M1−* is the performance without word formation features; 2) Line *M1+* is the performance when verb formation features are added. Line 4 to 8 shows the performance of the two-stage systems: 1) Line *M2−/AI* and *M2+/AI* shows the performance of AI phase without and within word formation features respectively; 2) Line *M2−/SRC* shows the SRC performance with trivial word-based features (i.e. frame features and verb formation features are not used); 3) Line *M2+wf/SRC* is the improved SRC performance when coarse verb formation features are added; 4) Line *M2+/SRC* is the SRC performance with all features; 5) Line *M2−AI+SRC* shows the performance of SRL system, which uses baseline features to identify arguments, and use all features to classify arguments.

6.4 Discussion

The results summarized in Table 4 indicate that according to the-state-of-the-art in Chinese parsing, SRL systems based on shallow parsing outperforms the ones based on full parsing. Comparison between *one-stage strategy* and *two-stage strategy* indicates 1) that there is no significant difference in the F-measure; and 2) that *two-stage strategy* method can achieve higher recall while *one-stage strategy* method can achieve higher precision. Both the *one-stage strategy* and *two-stage strategy* methods yield significant improvements over the best reported SRL performance in the literature, especially in terms of recall performance. Comparison SRL performance with full parses and partial parses indicates that both models have strong and weak points. The full parse based method can implement high precision SRL systems, while the partial parse based methods can implement high recall SRL systems. This is further justification for combination strategies that combine these independent SRL models.

Generally, Table 4 shows that verb formation features can enhance Chinese SRL, especially for fine-grained role classification. The effect of word formation in formation in both shallow parsing and SRL suggests that automatic word formation analyzing is very important for Chinese language processing. The rule-based algorithm is just a preliminary study on this new topic, which requires more research effort.

Num of words	P (%)	R (%)	$F_{\beta=1}$
Length = 1	84.69%	75.48%	79.82
Length = 2	82.14%	74.21%	77.97
Length = 3	75.43%	63.98%	69.24
Length = 4	75.71%	65.63%	70.32
Length = 5	72.46%	64.38%	68.18
Length = 6	72.97%	66.21%	69.43
Length = 7	77.03%	67.65%	72.04
Length = 8	74.39%	57.28%	64.72
Length = 9	66.67%	51.16%	57.89
Length = 10	68.08%	58.28%	62.80
Length = 11+	67.40%	57.71%	62.18

Table 5: SRL performance with arguments of different length

Though our SRC module does not use any parsing information, our system can achieve 93.42% accuracy, comparing the best gold parse based result 94.68% in the literature. This result suggests that Chinese SRC system, even without parsing, can reach a considerable good performance. The main reason is that in Chinese, arguments with different semantic types have discriminative boundary words, which can be extracted without parsing. It is very clear that the main bottleneck for Chinese SRL is to accurately identify arguments rather than to disambiguate their detailed semantic types.

Table 5 summarizes the labeling performance for argument of different length. It is not surprising that arguments are more and more difficult to rightly recognize as the increase of their length. But the performance decline slows up when the length of arguments is larger than 10. In other words, some of the arguments that are composed of many words can still be rightly identified. The main reason for this point is that these arguments usually have clear collocation words locating at argument boundaries. Take the sentences below for example,

3. 包括[*A*1 ..... 等] (including ... etc.)

the object of the verb "包括/include" has a definite collocation word "等/etc.", and therefore this object is easy to be recognized as a *A1*.

7 Conclusion

In this paper, we discuss Chinese SRL on the basis of partial syntactic structure. Our systems advance the state-of-the-art in Chinese SRL. We first

extend the study on Chinese shallow parsing and implement a good shallow parser. On the basis of partial parses, SRL are formulated as a sequence labeling problem, performing IOB2 decisions on the syntactic chunks of the sentence. We exploit a wide variety of features based on words, POS tags, and partial syntax. Additionally, we discuss a language special problem, i.e. Chinese word formation. Experimental results show that coarse word formation information can help shallow parsing, especially for NP-chunk recognition. A rule-based algorithm is put forward to automatically acquire Chinese verb formation, which is empirically shown to enhance SRL.

Acknowledgments

This work is supported by NSFC Project 60873156, 863 High Technology Project of China 2006AA01Z144 and the Project of Toshiba (China) R&D Center.

We would like to thank Weiwei Ding for his good advice on this research.

We would also like to thank the anonymous reviewers for their helpful comments.

References

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In Hwee Tou Ng and Ellen Riloff, editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 89–97, Boston, Massachusetts, USA, May 6 - May 7. Association for Computational Linguistics.

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. An empirical study of Chinese chunking. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 97–104, Sydney, Australia, July. Association for Computational Linguistics.

Yaodong Chen, Ting Wang, Huowang Chen, and Xishan Xu. 2008. Semantic role labeling of Chinese using transductive svm and semantic heuristics. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Weiwei Ding and Baobao Chang. 2008. Improving Chinese semantic role classification with hierarchical feature selection strategy. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 324–333, Honolulu, Hawaii, October. Association for Computational Linguistics.

Kadri Hacioglu and Wayne Ward. 2003. Target word detection and semantic role chunking using support vector machines. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 25–27, Morristown, NJ, USA. Association for Computational Linguistics.

S. Sathiya Keerthi and Dennis DeCoste. 2005. A modified finite newton method for fast solution of large scale linear svms. *J. Mach. Learn. Res.*, 6:341–361.

Taku Kudo and Yuji Matsumoto. 2000. Use of support vector learning for chunk identification. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning*, pages 142–144, Morristown, NJ, USA. Association for Computational Linguistics.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 217–220, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Sameer S. Pradhan, Wayne Ward, and James H. Martin. 2008. Towards robust semantic role labeling. *Comput. Linguist.*, 34(2):289–310.

L. A. Ramshaw and M. P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd ACL/SIGDAT Workshop on Very Large Corpora, Cambridge, Massachusetts, USA*, pages 82–94.

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantc parsing of Chinese. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere Comas. 2007. Combination strategies for semantic role labeling. *J. Artif. Intell. Res. (JAIR)*, 29:105–151.

Nianwen Xue and Martha Palmer. 2005. Automatic semantic role labeling for Chinese verbs. In *in Proceedings of the 19th International Joint Conference on Artificial Intelligence*, page 2005.

Nianwen Xue. 2008. Labeling chinese predicates with semantic roles. *Computational Linguistics*, 34(2):225–255.

Discovery of Term Variation in Japanese Web Search Queries

Hisami Suzuki, Xiao Li, and Jianfeng Gao
Microsoft Research, Redmond
One Microsoft Way, Redmond, WA 98052 USA
{hisamis,xiaol,jfgao}@microsoft.com

Abstract

In this paper we address the problem of identifying a broad range of term variations in Japanese web search queries, where these variations pose a particularly thorny problem due to the multiple character types employed in its writing system. Our method extends the techniques proposed for English spelling correction of web queries to handle a wider range of term variants including spelling mistakes, valid alternative spellings using multiple character types, transliterations and abbreviations. The core of our method is a statistical model built on the MART algorithm (Friedman, 2001). We show that both string and semantic similarity features contribute to identifying term variation in web search queries; specifically, the semantic similarity features used in our system are learned by mining user session and click-through logs, and are useful not only as model features but also in generating term variation candidates efficiently. The proposed method achieves 70% precision on the term variation identification task with the recall slightly higher than 60%, reducing the error rate of a naïve baseline by 38%.

1 Introduction

Identification of term variations is fundamental to many NLP applications: words (or more generally, terms) are the building blocks of NLP applications, and any robust application must be able to handle variations in the surface representation of terms, be it a spelling mistake, valid spelling variation, or abbreviation. In search applications, term variations can be used for query expansion, which generates additional query terms for better matching with the terms in the document set. Identifying term variations is also useful in other scenarios where semantic equivalence of terms is sought, as it represents a very special case of paraphrase.

This paper addresses the problem of identifying term variations in Japanese, specifically for the purpose of query expansion in web search, which appends additional terms to the original query string for better retrieval quality. Query expansion has been shown to be effective in improving web search results in English, where different methods of generating the expansion terms have been attempted, including relevance feedback (e.g., Salton and Buckley, 1990), correction of spelling errors (e.g., Cucerzan and Brill, 2004), stemming or lemmatization (e.g., Frakes, 1992), use of manually- (e.g., Aitchison and Gilchrist, 1987) or automatically- (e.g., Rasmussen 1992) constructed thesauri, and Latent Semantic Indexing (e.g., Deerwester et al, 1990). Though many of these methods can be applied to Japanese query expansion, there are unique problems posed by Japanese search queries, the most challenging of which is that valid alternative spellings of a word are extremely common due to the multiple script types employed in the language. For example, the word for 'protein' can be spelled as たんぱくしつ, タンパク質, 蛋白質, たん白質 and so on, all pronounced *tanpakushitsu* but using combinations of different script types. We give a detailed description of the problem posed by the Japanese writing system in Section 2. Though there has been previous work on addressing specific subsets of spelling alterations within and across character types in Japanese, there has not been any comprehensive solution for the purpose of query expansion.

Our approach to Japanese query expansion is unique in that we address the problem comprehensively: our method works independently of the character types used, and targets a wide range of term variations that are both orthographically and semantically similar, including spelling errors, valid alternative spellings, transliterations and abbreviations. As described in Section 4, we define the problem of term variation identifica-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1484–1492,
Singapore, 6-7 August 2009.

tion as a binary classification task, and build two types of classifiers according to the maximum entropy model (Berger et al., 1996) and the MART algorithm (Friedman, 2001), where all term similarity metrics are incorporated as features and are jointly optimized. Another important contribution of our approach is that we derive our semantic similarity models by mining user query logs, which has been explored for the purposes of collecting related words (e.g., Jones et al., 2006a), improving search results ranking (e.g., Craswell and Szummer, 2007) and learning query intention (e.g., Li et al., 2008), but not for the task of collecting term variations. We show that our semantic similarity models are not only effective in the term variation identification task, but also for generating candidates of term variations much more efficiently than the standard method whose candidate generation is based on edit distance metrics.

2 Term Variations in Japanese

In this section we give a summary of the Japanese writing system and the problem it poses for identifying term variations, and define the problem we want to solve in this paper.

2.1 The Japanese Writing System

There are four different character types that are used in Japanese text: *hiragana*, *katakana*, *kanji* and Roman alphabet. Hiragana and katakana are the two subtypes of *kana* characters, which are syllabic character sets, each with about 50 basic characters. There is a one-to-one correspondence between hiragana and katakana characters, and, as they are phonetic, they can be unambiguously converted into a sequence of Roman characters. For example, the word for 'mackerel' is spelled in hiragana as さば or in katakana as サバ, both of which can be transcribed in Roman characters as *saba*, which is how the word is pronounced. Kanji characters, on the other hand, are ideographic and therefore numerous – more than 5,000 are in common usage. One difficulty in handling Japanese kanji is that each character has multiple pronunciations, and the correct pronunciation is determined by the context in which the character is used. For instance, the character 行 is read as *kou* in the word 銀行 *ginkou* 'bank', *gyou* in 行 'column', and *i* or *okona* in 行った *itta* 'went' or *okonatta* 'done' depending on the context in which the word is used.[1] Proper name readings are particularly difficult to disambiguate, as their pronunciation cannot be inferred from the context (they tend to have the same grammatical function) or from the dictionary (they tend to be out-of-vocabulary). Therefore, in Japanese, computing a pronunciation-based edit distance metric is not straightforward, as it requires estimating the readings of kanji characters.

2.2 Term Variation by Character Type

Spelling variations are commonly observed both within and across character types in Japanese. Within a character type, the most prevalent is the variation observed in katakana words. Katakana is used to transliterate words from English and other foreign languages, and therefore reflects the variations in the sound adaptation from the source language. For example, the word 'spaghetti' is transliterated into six different forms (スパゲッティ *supagetti*, スパゲッティー *supagettii*, スパゲッテイ *supagettei*, スパゲティ *supageti*, スパゲティー *supagetii*, スパゲテイ *supagetei*) within a newspaper corpus (Masuyama et al., 2004).

Spelling variants are also prevalent across character types: in theory, a word can be spelled using any of the character types, as we have seen in the example for the word 'protein' in Section 1. Though there are certainly preferred character types for spelling each word, variations are still very common in Japanese text and search queries. Alterations are particularly common among hiragana, katakana and kanji (e.g. さば~サバ~ 鯖 *saba* 'mackerel'), and between katakana and Roman alphabet (e.g. フェデックス *fedekkusu* fedex). This latter case constitutes the problem of transliteration, which has been extensively studied in the context of machine translation (e.g. Knight and Graehl, 1998; Bilac and Tanaka, 2004; Brill et al., 2001).

2.3 Term Variation by Re-write Categories

Table 1 shows the re-write categories of related terms observed in web query logs, drawing on our own data analysis as well as on previous work such as Jones et al. (2006a) and Okazaki et al. (2008b). Categories 1 though 9 represent strictly synonymous relations; in addition, terms in Categories 1 through 5 are also similar orthographically or in pronunciation. Categories 10

[1] In a dictionary of 200K entries, we find that on average each kanji character has 2.5 readings, with three characters (直,生,空) with as many as 11 readings.

Categories	Example in English	Example in Japanese
1. Spelling mistake	aple ~ apple	グウグル *guuguru* ~ グーグル *gu-guru* 'google'
2. Spelling variant	color ~ colour	さば~サバ~鯖; スパゲティ~スパゲッティー (Cf. Sec.2.2)
3. Inflection	matrix ~ matrices	作る *tsukuru* 'make' ~ 作った *tsukutta* 'made'
4. Transliteration		フェデックス ~ fedex 'Fedex'
5. Abbreviation/ Acronym	macintosh ~ mac	世界銀行 *sekaiginkou* ~ 世銀 *segin* 'World Bank'; マクドナルド *makudonarudo* ~ マック *makku* 'McDonald's'
6. Alias	republican party ~ gop	フランス *furansu* ~ 仏 *futsu* 'France'
7. Translation	パキスタン大使館 *pakisutantaishikan* ~ Pakistan embassy	
8. Synonym	carcinoma ~ cancer	暦 *koyomi* ~ カレンダー *karendaa* 'calendar'
9. Abbreviation (user specific)	mini ~ mini cooper	クロネコヤマト *kuronekoyamato* ~ クロネコ *kuroneko* (name of a delivery service company)
10. Generalization	nike shoes ~ shoes	シビック 部品 *shibikku buhin* 'Civic parts' ~ 車 部品 *kuruma buhin* 'car parts'
11. Specification	ipod ~ ipod nano	東京駅 *toukyoueki* 'Tokyo station' ~ 東京駅時刻表 *toukyouekijikokuhyou* 'Tokyo station timetable'
12. Related	windows ~ microsoft	トヨタ *toyota* 'Toyota' ~ ホンダ *honda* 'Honda'

Table 1: Categories of Related Words Found in Web Search Logs

through 12, on the other hand, specify non-synonymous relations.

Different sets out of these categories can be useful for different purposes. For example, Jones et al (2006a; 2006b) target all of these categories, as their goal is to collect related terms as broadly as possible for the application of sponsored search, i.e., mapping search queries to a small corpus of advertiser listings. Okazaki et al. (2008b) define their task narrowly, to focusing on spelling variants and inflection, as they aim at building lexical resources for the specific domain of medical text.

For web search, a conservative definition of the task as dealing only with spelling errors has been successful for English; a more general definition using related words for query expansion has been a mixed blessing as it compromises retrieval precision. A comprehensive review on this topic is provided by Baeza-Yates and Ribeiro-Neto (1999). In this paper, therefore, we adopt a working definition of the term variation identification task as including Categories 1 through 5, i.e., those that are synonymous and also similar in spelling or in pronunciation.[2] This definition is reasonably narrow so as to make automatic discovery of term variation pairs realistic, while covering all common cases of term variation in Japanese, including spelling variants and transliterations. It is also appropriate for the purpose of query expansion: because term variation defined in this manner is based on spelling or pronunciation similarity, their meaning and function tend to be completely equivalent, as opposed to Categories 6 through 9, where synonymy is more context- or user-dependent. This will ensure that the search results by query expansion will avoid the problem of compromised precision.

3 Related Work

In information retrieval, the problem of vocabulary mismatch between the query and the terms in the document has been addressed in many ways, as mentioned in Section 1, achieving varying degrees of success in the retrieval task. In particular, our work is closely related to research in spelling correction for English web queries (e.g., Cucerzan and Brill, 2004; Ahmad and Kondrak, 2005; Li et al., 2006; Chen et al., 2007). Among these, Li et al. (2006) and Chen et al. (2007) incorporate both string and semantic similarity in their discriminative models of spelling correction, similarly to our approach. In Li et al. (2006), semantic similarity was computed as distributional similarity of the terms using query strings in the log as context. Chen et al. (2007) point out that this method suffers from the data sparseness problem in that the statistics for rarer terms are unreliable, and propose using web search results as extended contextual information. Their method, however, is expensive as it requires web search results for each query-candidate pair, and also because their candidate set, generated using an edit distance function and phonetic similarity from query log data, is impractically large and must be pruned by using a language model. Our approach differs from these methods in that we exploit user query logs to derive semantic knowledge of terms, which is

[2] In reality, Category 3 (Inflection) is extremely rare in Japanese web queries, because nouns do not inflect in Japanese, and most queries are nominals.

used both for the purpose of generating a candidate set efficiently and as features in the term variation identification model.

Acquiring semantic knowledge from a large quantity of web query logs has become popular in recent years. Some use only query strings and their counts for learning word similarity (e.g., Sekine and Suzuki, 2007; Komachi and Suzuki, 2008), while others use additional information, such as the user session information (i.e., a set of queries issued by the same user within a time frame, e.g., Jones et al., 2006a) or the URLs clicked as a result of the query (e.g., Craswell and Szummer, 2007; Li et al., 2008). This additional data serves as an approximation to the meaning of the query; we use both user session and click-through data for discovering term variations.

Our work also draws on some previous work on string transformation, including spelling normalization and transliteration. In addition to the simple Levenshtein distance, we also use generalized string-to-string edit distance (Brill and Moore, 2000), which we trained on aligned katakana-English word pairs in the same manner as Brill et al. (2001). As mentioned in Section 2.2, our work also tries to address the individual problems targeted by such component technologies as Japanese katakana variation, English-to-katakana transliteration and katakana-to-English back-transliteration in a unified framework.

4 Discriminative Model of Identifying Term Variation

Recent work in spelling correction (Ahmed and Kondrak, 2005; Li et al., 2006; Chen et al., 2007) and normalization (Okazaki et al., 2008b) formulates the task in a discriminative framework:

$$c^* = \text{argmax}_{c \in \text{gen}(q)} P(c|q)$$

This model consists of two components: gen(q) generates a list of candidates $C(q)$ for an input query q, which are then ranked by the ranking function $P(c|q)$. In previous work, gen(q) is typically generated by using an edit distance function or using a discriminative model trained for its own purpose (Okazaki et al., 2008b), often in combination with a pre-complied lexicon. In the current work, we generate the list of candidates by learning pairs of queries and their re-write candidates automatically from query session and click logs, which is far more robust and efficient than using edit distance functions. We describe our candidate generation method in detail in Section 5.1.

Unlike the spelling correction and normalization tasks, our goal is to *identify* term variations, i.e., to determine whether each query-candidate pair (q,c) constitutes a term variation or not. We formulate this problem as a binary classification task. There are various choices of classifiers for such a task: we chose to build two types of classifiers that make a binary decision based on the probability distribution $P(c|q)$ over a set of feature functions $f_i(q,c)$. In maximum entropy framework, this is defined as:

$$P(c|q) = \frac{\exp \sum_{i=1}^{K} \lambda_i f_i(c,q)}{\sum_c \exp \sum_{i=1}^{K} \lambda_i f_i(c,q)}$$

where $\lambda_1,\ldots, \lambda_k$ are the feature weights. The optimal set of feature weights λ^* is computed by maximizing the log-likelihood of the training data. We used stochastic gradient descent for training the model with a Gaussian prior.

The second classifier is built on MART (Friedman, 2001), which is a boosting algorithm. At each boosting iteration, MART builds a regression tree to model the functional gradient of the cost function (which is cross entropy in our case), evaluated on all training samples. MART has three main parameters: M, the total number of boosting iterations, L, the number of leaf nodes for each regression tree, and v, the learning rate. The optimal values of these parameters can be chosen based on performance on a validation set. In our experiments, we found that the performance of the algorithm is relatively insensitive to these parameters as long as they are in a reasonable range: given the training set of a few thousand samples or more, as in our experiments, M=100, L=15, and v=0.1 usually give good performance. Smaller trees and shrinkage may be used if the training data set is smaller.

The classifiers output a binary decision according to $P(c|q)$: positive when $P(c|q) > 0.5$ and negative otherwise.

5 Experiments

5.1 Candidate Generation

We used a set of Japanese query logs collected over one year period in 2007 and 2008. More specifically, we used two different extracts of log data for generating term variation candidate pairs:

Query session data. From raw query logs, we extracted pairs of queries q_1 and q_2 such that they are (i) issued by the same user; (ii) q_2 follows within 3 minutes of issuing q_1; and (iii) q_2 generated at least one click of a URL on the result

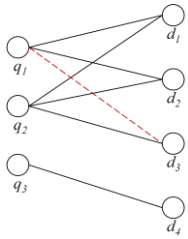


Figure 1. Random Walk Algorithm

page while q_1 did not result in any click. We then scored each query pair (q_1,q_2) in this subset using the log-likelihood ratio (LLR, Dunning, 1993) between q_1 and q_2, which measures the mutual dependence within the context of web search queries (Jones et al., 2006a). After applying an LLR threshold (LLR > 15) and a count cutoff (we used only the top 15 candidate q_2 according to the LLR value for each q_1), we obtained a list of 47,139,976 pairs for the 14,929,497 distinct q_1, on average generating 3.2 candidates per q_1[3]. We took this set as comprising query-candidate pairs for our model, along with the set extracted by click-through data mining explained below.

Click-through data. This data extract is based on the idea that if two queries led to the same URLs being repeatedly clicked, we can reasonably infer that the two queries are semantically related. This is similar to computing the distributional similarity of terms given the context in which they appear, where context is most often defined as the words co-occurring with the terms. Here, the clicked URLs serve as their context.

One challenge in using the URLs as contextual information is that the contextual representation in this format is very sparse, as user clicks are rare events. To learn query similarities from incomplete click-through data, we used the random walk algorithm similar to the one described in Craswell and Szummer (2007). Figure 1 illustrates the basic idea: initially, document d_3 has a click-through link consisting of query q_2 only; the random walk algorithm adds the link from d_3 to q_1, which has a similar click pattern as q_2. Formally, we construct a click graph which is a bipartite-graph representation of click-through data. We use $\{q_i\}_{i=1}^{m}$ to represent a set of query nodes and $\{d_j\}_{j=1}^{n}$ a set of document nodes. We further define an $m \times n$ matrix W in which element W_{ij} represents the click count associated with (q_i, d_j). This matrix can be normalized to be a query-to-document transition matrix, denoted by A, where $A_{ij} = p^{(1)}(d_j|q_i)$ is the probability that q_i transits to d_j in one step. Similarly, we can normalize the transpose of W to be a document-to-query transition matrix, denoted by B, where $B_{j,i} = p^{(1)}(q_i|d_j)$. It is easy to see that using A and B we can compute the probability of transiting from any node to any other node in k steps. In this work, we use a simple measure which is the probability that one query transits to another in two steps, and the corresponding probability matrix is given by AB.

We used this probability and ranked all pairs of queries in the same raw query logs as in the query session data described above to generate additional candidates for term variation pairs. 20,308,693 pairs were extracted after applying the count cutoff of 5, generating on average 6.8 candidates for 2,973,036 unique queries.

It is interesting to note that these two data extracts are quite complementary: of all the data generated, only 4.2% of the pairs were found in both the session and click-through data. We believe that this diversity is attributable to the nature of the extracts: the session data tends to collect the term pairs that are issued by the same user as a result of conscious re-writing effort, such as typing error corrections and query specifications (Categories 1 and 11 in Table 1), while the click-though data collects the terms issued by different users, possibly with different intentions, and tends to include many spelling variants, synonyms and queries with different specificity (Categories 2, 8, 10 and 11).

5.2 Features

We used the same set of features for the maximum entropy and MART models, which are given in Table 2. They are divided into three main types: string similarity features (1-16), semantic similarity features (17, 18), and character type features (19-39). Among the string similarity features, half of them are based on Levenshtein distance applied to surface forms (1-8), while the other half is based on Levenshtein and string-to-string edit distance metrics computed over the Romanized form of the query, reflecting its pronunciation. The conversion into Roman characters was done deterministically for kana characters using a simple mapping table. For Romanizing kanji characters, we used the function available from Windows IFELanguage API (version

[3] We consider each query as an unbreakable term in this paper, so term variation is equivalent to query variation.

String similarity features (16 real-valued features)
1. Lev distance on surface form
2. Lev distance on surface form normalized by *q1* length
3. Lev distance on surface form using character equivalence table
4. Lev distance on surface form normalized by *q1* length using character equivalence table
5. Lev distance on surface form w/o space
6. Lev distance on surface form normalized *q1* length w/o space
7. Lev distance on surface form using character equivalence table w/o space
8. Lev distance on surface form normalized by *q1* using character equivalence table w/o space
9. Lev distance on Roman
10. Lev distance on Roman normalized by *q1* length
11. Alpha-beta edit distance on Roman
12. Alpha-beta edit distance on Roman normalized by *q1* length
13. Lev distance on Roman w/o space
14. Lev distance on Roman normalized by *q1* length w/o space
15. Alpha-beta edit distance on Roman w/o space
16. Alpha-beta edit distance on Roman normalized by *q1* length w/o space
Features for semantic similarity (2 real-valued features)
17. LLR score
18. Click-though data probability
Character type features (21 binary features)
19. BothHira, 20. BothKata, 21. BothRoman, 22. BothKanji, 23. BothMixedNoKanji, 24. BothMixed, 25. HiraKata, 26. HiraKanji, 27. HiraRoman, 28. HiraMixedNoKanji, 29. HiraMixed, 30. KataKanji, 31.KataRoman, 32. KataMixedNoKanji, 33. KataMixed, 34. KanjiRoman, 35. KanjiMixedNoKanji, 36. KanjiMixed, 37. RomanMixedNoKanji, 38. RomanMixed, 39. MixedNoKanjiMixed

Table 2: Classifier Features

2).[4] The character equivalence table mentioned in the features 3,4,7,8 is a table of 643 pairs of characters that are known to be equivalent, including kanji allography (same kanji in different graphical styles). The alpha-beta edit distance (11, 12, 15, 16) is the string-to-string edit distance proposed in Brill and Moore (2001), which we trained over about 60K parallel English-to-katakana Wikipedia title pairs, specifically to capture the edit operations between English and katakana words, which are different from the edit operations between two Japanese words. Semantic similarity features (17, 18) use the LLR score from the session data, and the click-though pair probability described in the subsection above. Finally, features 19-39 capture the script types of the query-candidate pair. We first defined six basic character types for each query or candidate: Hira (hiragana only), Kata (katakana only), Kanji (kanji only), Roman (Roman alphabet only), MixedNoKanji (includes more than one character sets but not kanji) and Mixed (includes more than one character sets with kanji). We then derived 21 binary features by concatenating these basic character type features for the combination of query and candidate strings. For example, if both the query and candidate are in hiragana, BothHira will be on; if the query is Mixed and the candidate is Roman, then RomanMixed will be on. Punctuation characters and Arabic numerals were treated as being transparent to character type assignment. The addition of these features is motivated by the assumption that appropriate types of edit distance operations might depend on different character types for the query-candidate pair.

Since the dynamic ranges of different features can be drastically different, we normalized each feature dimension to a normal variable with zero-mean and unit-variance. We then used the same normalized features for both the maximum entropy and the MART classifiers.

5.3 Training and Evaluation Data

In order to generate the training data for the binary classification task, we randomly sampled the query session (5,712 samples) and click-through data (6,228 samples), and manually labeled each pair as positive or negative: the positive label was assigned when the term pair fell into Categories 1 through 5 in Table 1; otherwise it was assigned a negative label. Only 364 (6.4%) and 244 (3.9%) of the samples were positive examples for the query session and click-through data respectively, which makes the baseline per-

[4] http://msdn.microsoft.com/en-us/library/ms970129.aspx. We took the one-best conversion result from the API. The conversion accuracy on a randomly sampled 100 kanji queries was 89.6%.

formance of the classifier quite high (always predict the negative label – the accuracy will be 95%). Note, however, that these data sets include term variation candidates much more efficiently than a candidate set generated by the standard method that uses an edit distance function with a threshold. For example, there is a query-candidate pair *q*=家風情報 *kafuujouhou* 'house-style information' *c*= 花 粉 情 報 *kafunjouhou* 'pollen information') in the session data extract, the first one of which is likely to be a misspelling of the second.[5] If we try to find candidates for the query 家風情報 using an edit distance function naively with a threshold of 2 from the queries in the log, we end up collecting a large amount of completely irrelevant set of candidates such as 台風情報 *taifuujouhou* 'typhoon information', 株情報 *kabu jouhou* 'stock information', 降雨情報 *kouu jouhou* 'rainfall information' and so on – as many as 372 candidates were found in the top one million most frequent queries in the query log from the same period; for rarer queries these numbers will only be worse. Computing the edit distance based on the pronunciation will not help here: the examples above are within the edit distance of 2 even in terms of Romanized strings.

Another advantage of generating the annotated data using the result of query log data mining is that the annotation process is less prone to subjectivity than creating the annotation from scratch. As Cucerzan and Brill (2004) point out, the process of manually creating a spelling correction candidate is seriously flawed as the intention of the original query is completely lost: for the query *gogle*, it is not clear out of context if the user meant *goggle*, *google*, or *gogle*. Using data mined from query logs solves this problem: an annotator can safely assume that if *gogle-goggle* appears in the candidate set, it is very likely to be a valid term variation intended by the user. This makes the annotation more robust and efficient: the inter-annotator agreement rate for 2,000 query pairs by two annotators was 95.7% on our data set, each annotator spending only about two hours to annotate 2,000 pairs.

5.4 Results and Discussion

In order to compare the performance of two classifiers, we first built maximum entropy and MART classifiers as described in Section 4 using all the features in Section 5.2. We run five experiments using different random split of training and test data: in each run, we used 10,000 samples for training and the remaining 1,940 samples for testing, and measured the performance of the two classifiers on the task of term variation identification in terms of the error rate i.e., 1–accuracy. The results, average over five runs, were 4.18 for the maximum entropy model, and 3.07 for the MART model. In all five runs, the MART model outperformed the maximum entropy classifier. This is not surprising given the superior performance of tree-boosting algorithms previously reported on similar classification tasks (e.g., Hastie et al., 2001). In our task where different types of features are likely to perform better when they are combined (such as semantic features and character types features), MART would be a better fit than linear classifiers because the decision trees generated by MART optimally combines features in the local sense. In what follows, we only discuss the results produced by MART for further experiments. Note that the baseline classifier, which always predicts the label to be negative, achieves 95.04% in accuracy (or 4.96% error rate), which sounds extremely high, but in fact this baseline classifier is useless for the purpose of collecting term variations, as it learns none of them by classifying all samples as negative.

Features	Error rate (%)
A. All features (1-39 in Table 2)	3.07
B. String features only (1-16)	3.49
C. Surface string features only (1-8)	4.9
D. No semantic feats (1-16,19-39)	3.28
E. No character type feats (1-18)	3.5

Table 3: Results of Features Ablation Experiments Using MART Model

For evaluating the contribution of different types of features in Section 5.2, we performed feature ablation experiments using MART. Table 3 shows the results in error rate by various MART classifiers using different combination of features. The results in this table are also averaged over five run with random training/test data split. From Table 3, we can see that the best performance was achieved by the model using all features (line A of the table), which reduces the baseline error rate (4.96%) by 38%. The improvement is statistically significant according to the McNemar test ($P < 0.05$). Models that use string edit distance features only (lines B and C) did not perform well: in particular, the model that uses surface edit distance features only

[5] 家風情報 does not make any sense in Japanese; on the other hand, information about cedar pollen is commonly sought after in spring due to widespread pollen allergy.

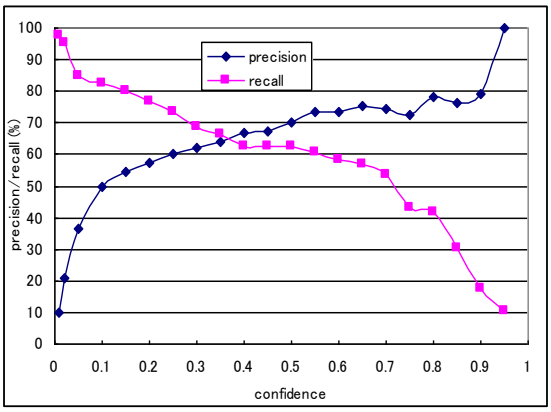


Figure 2: Precision/Recall Curve of MART

without considering the term pronunciation performed horribly (line C), which confirms the results reported by Jones et al. (2006b). However, unlike Jones et al. (2006b), we see a positive contribution of semantic features: the use of semantic features reduced the error rate from 3.28 (line D) to 3.07 (line A), which is statistically significant. This may be attributable to the nature of semantic information used in our experiments: we used the user session and click-though data to extract semantic knowledge, which may be semantically more specific than the probability of word substitution in a query collection as a whole, which is used by Jones et al. (2006b). Finally, the character type features also contributed to reducing the error rate (lines A and E). In particular, the observation that the addition of semantic features without the character type features (comparing lines B and E) did not improve the error rate indicates that the character type features are also important in bringing about the contribution of semantic features.

Figure 2 displays the test data precision/recall curve of one of the runs of MART that uses all features. The x-axis of the graph is the confidence score of classification $P(c|q)$, which was set to 0.5 for the results in Table 3. At this confidence, the model achieves 70% precision with the recall slightly higher than 60%. In the graph, we observe a familiar trade-off between precision and recall, which is useful for practical applications that may favor one over the other.

In order to find out where the weaknesses of our classifiers lie, we performed a manual error analysis on the same MART run whose results are shown in Figure 2. Most of the classification errors are false negatives, i.e., the model failed to predict a case of term variation as such. The most conspicuous error is the failure to capture abbreviations, such as failing to capture the alteration between 十条中学校 *juujoochuugakkou* 'Juujoo middle school' and 十条中 *juujoochuu*, which our edit distance-based features fail as the length difference between a term and its abbreviation is significant. Addition of more targeted features for this subclass of term variation (e.g., Okazaki et al., 2008a) is called for, and will be considered in future work. Mistakes in the Romanization of kanji characters were not always punished as the query and the candidate string may contain the same mistake, but when they occurred in either in the query or the candidate string (but not in both), the result was destructive: for example, we assigned a wrong Romanization on 水銀燈 as *suiginnakari* 'mercury lamp', as opposed to the correct *suiginntou*, which causes the failure to capture the alteration with 水銀痘 *suiginntou*, (a misspelling of 水銀燈). Using N-best (N>1) candidate pronunciations for kanji terms or using all possible pronunciations for kanji characters might reduce this type of error. Finally, the features of our models are the edit distance functions themselves, rather than the individual edit rules or operations. Using these individual operations as features in the classification task directly has been shown to perform well on spelling correction and normalization tasks (e.g., Brill and Moore, 2000; Okazaki et al., 2008b). Okazaki et al.'s (2008b) method of generating edit operations may not be viable for our purposes, as they assume that the original and candidate strings are very similar in their surface representation – they target only spelling variants and inflection in English. One interesting future avenue to consider is to use the edit distance functions in our current model to select a subset of query-candidate pairs that are similar in terms of these functions, separately for the surface and Romanized forms, and use this subset to align the character strings in these query-candidate pairs as described in Brill and Moore (2000), and add the edit operations derived in this manner to the term variation identification classifier as features.

6 Conclusion

In this paper we have addressed the problem of acquiring term variations in Japanese query logs for the purpose of query expansion. We generate term variation candidates efficiently by mining query log data, and our best classifier, based on the MART algorithm, can make use of both edit-distance-based and semantic features, and can identify term variation with the precision of 70% at the recall slightly higher than 60%. Our next

goal is to use and evaluate the term variation collected by the proposed method in an actual search scenario, as well as improving the performance of our classifier by using individual, character-dependent edit operations as features in classification.

References

Ahmad, Farooq, and Grzegorz Kondrak. 2005. Learning a spelling error model from search query logs. In *Proceedings of EMNLP*, pp.955-962.

Aitchison, J. and A. Gilchrist. 1987. *Thesaurus Construction: A Practical Manual*. Second edition. ASLIB, London.

Aramaki, Eiji, Takeshi Imai, Kengo Miyo, and Kazuhiko Ohe. 2008. Orthographic disambiguation incorporating transliterated probability. In *Proceedings of IJCNLP*, pp.48-55.

Baeza-Yates, Ricardo, and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison Wesley.

Berger, A.L., S. A. D. Pietra, and V. J. D. Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1): 39-72.

Bilac, Slaven, and Hozumi Tanaka. 2004. A hybrid back-transliteration system for Japanese. In *Proceedings of COLING*, pp.597-603.

Brill, Eric, Gary Kacmarcik and Chris Brockett. 2001. Automatically harvesting katakana-English term pairs from search engine query logs. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium* (NLPRS-2001), pp.393-399.

Brill, Eric, and Robert C. Moore. 2000. An improved error model for noisy channel spelling. In *Proceedings of ACL*, pp.286-293.

Chen, Qing, Mu Li and Ming Zhou. 2007. Improving query spelling correction using web search results. In *Proceedings of EMNLP-CoNLL*, pp.181-189.

Craswell, Nick, and Martin Szummer. 2007. Random walk on the click graph. In *Proceedings of SIGIR*.

Cucerzan, Silviu, and Eric Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP*, pp.293-300.

Deerwester, S., S.T. Dumais, T. Landauer and Harshman. 1990. Indexing by latent semantic analysis. In *Journal of the American Society for Information Science*, 41(6): 391-407.

Dunning, Ted. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1): 61-74.

Frakes, W.B. 1992. Stemming algorithm. In W.B.Frakes and R.Baeza-Yates (eds.), *Information Retrieval: Data Structure and Algorithms*, Chapter 8. Prentice Hall.

Friedman, J. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5).

Jones, Rosie, Benjamin Rey, Omid Madani and Wiley Greiner. 2006a. Generating query substitutions. In *Proceedings of WWW*, pp.387–396.

Jones, Rosie, Kevin Bartz, Pero Subasic and Benjamin Rey. 2006b. Automatically generating related aueries in Japanese. *Language Resources and Evaluation* 40: 219-232.

Hastie, Trevor, Robert Tibshirani and Jerome Friedman. 2001. *The Elements of Statistical Learning.* Springer.

Knight, Kevin, and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4): 599-612.

Komachi, Mamoru and Hisami Suzuki. 2008. Minimally supervised learning of semantic knowledge from query logs. In *Proceedings of IJCNLP*, pp.358–365.

Li, Mu, Muhua Zhu, Yang Zhang and Ming Zhou. 2006. Exploring distributional similarity based models for query spelling correction. In *Proceedings of COLING/ACL*, pp.1025-1032.

Li, Xiao, Ye-Yi Wang and Alex Acero. 2008. Learning query intent from regularized click graphs. In *Proceedings of SIGIR*.

Masuyama, Takeshi, Satoshi Sekine, and Hiroshi Nakagawa. 2004. Automatic construction of Japanese katakana variant list from large corpus. In *Proceedings COLING*, pp.1214-1219.

Okazaki, Naoaki, Mitsuru Ishizuka and Jun'ichi Tsujii. 2008a. A discriminative approach to Japanese abbreviation extraction. In *Proceedings of IJCNLP*.

Okazaki, Naoaki, Yoshimasa Tsuruoka, Sophia Ananiadou and Jun'ichi Tsujii. 2008b. A discriminative candidate generator for string transformations. In *Proceedings of EMNLP*.

Rasmussen, E. 1992. Clustering algorithm. In W.B.Frakes and R.Baeza-Yates (eds.), *Information Retrieval: Data Structure and Algorithms*, Chapter 16. Prentice Hall.

Salton, G., and C. Buckley. 1990. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4): 288-297.

Sekine, Satoshi, and Hisami Suzuki. 2007. Acquiring ontological knowledge from query logs. In *Proceedings of WWW*, pp.1223-1224

Towards Discipline-Independent Argumentative Zoning: Evidence from Chemistry and Computational Linguistics

Simone Teufel
Computer Laboratory
Cambridge University
sht25@cl.cam.ac.uk

Advaith Siddharthan
Computer Laboratory
Cambridge University
as372@cl.cam.ac.uk

Colin Batchelor
Royal Society of Chemistry
Cambridge, UK
batchelorc@rsc.org

Abstract

Argumentative Zoning (AZ) is an analysis of the argumentative and rhetorical structure of a scientific paper. It has been shown to be reliably used by independent human coders, and has proven useful for various information access tasks. Annotation experiments have however so far been restricted to one discipline, computational linguistics (CL). Here, we present a more informative AZ scheme with 15 categories in place of the original 7, and show that it can be applied to the life sciences as well as to CL. We use a domain expert to encode basic knowledge about the subject (such as terminology and domain specific rules for individual categories) as part of the annotation guidelines. Our results show that non-expert human coders can then use these guidelines to reliably annotate this scheme in two domains, chemistry and computational linguistics.

1 Introduction

Teufel et al. (1999) define the task of Argumentative Zoning (AZ) as a sentence-by-sentence classification with mutually exclusive categories from the annotation scheme given in Fig. 1. The reasoning behind the categories is inspired by the notion of a *knowledge claim* (Myers, 1992; Luukkonen, 1992): the act of writing a paper corresponds to an attempt of claiming ownership for a new piece of knowledge, which is to be integrated into the repository of scientific knowledge in the authors' field by the process of peer review and publication. In the cause of this process, the authors have to convince the reviewers that the knowledge claim of the paper is valid (Swales, 1990; Hyland, 1998). What AZ aims to model, then, are some of the relevant stages in this argument. We divide the paper into *zones*, OTHER, OWN and BACKGROUND. These are defined on the basis of who owns the knowledge claim in the corresponding segment. There are also two categories which are defined by their relationship to existing work, BASIS and CONTRAST. That means that parts of the AZ scheme are similar to citation function classification schemes from the area of citation content analysis (Garfield, 1965; Weinstock, 1971; Spiegel-Rüsing, 1977), and to automatic citation function classification (Nanba and Okumura, 1999; Garzone and Mercer, 2000; Teufel et al., 2006). The remaining categories, AIM and TEXTUAL, fulfil different rhetorical functions for the presentation of the paper. AIM points out the paper's main knowledge claim, a rhetorical move which may be repeated in the conclusion and the introduction. TEXTUAL explains the physical location of information, e.g., by giving a section overview or presenting a summary of a subsection. On the basis of human-annotated training material, AZ can be automatically classified using supervised machine learning.

Category	Description
AIM	Statement of research goal.
BACKGROUND	Description of generally accepted background knowledge.
BASIS	Existing KC provides basis for new KC.
CONTRAST	An existing KC is contrasted, compared, or presented as weak.
OTHER	Description of existing KC.
OWN	Description of any other aspect of new KC.
TEXTUAL	Indication of paper's textual structure.

Figure 1: AZ Annotation Scheme (Teufel et al. 1999).

Rhetorical information marking is useful for

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1493–1502,
Singapore, 6-7 August 2009.

many novel information access tasks. For instance, information retrieval can profit from rhetorical information in the form of paradigm shift statements (Chichester et al., 2005), as papers containing such statements have a high impact in an area. 75% of the "Faculty of 1000 Biology" papers (which are chosen by experts for their special importance) contain paradigm shift sentences (Agnes Sandor, personal communication).

AZ annotation allows the construction of multi- and single document summaries which concentrate on differences and similarities to related (cited) work. AZ can also be used for search in a data base of scientific articles, in particular for enhanced citation indexing. This has been previously explored in a task-based evaluation, were users were asked to list positive and negative citations they would expect in a paper, given a short extract (Teufel, 2001). In that task, AZ-based extracts outperformed other document surrogates.

Feltrim et al. (2005) present a writing support system which analyses students' drafts of summaries for their PhD theses, performs an AZ analysis on them and critiques the rhetorical structure of the students' draft on the basis of it.

The definition of the AZ categories is based on rhetorical principles and should be decidable, in principle, without specific domain knowledge about what is discussed in detail in the paper. We present here the first evidence that AZ categories can be reliably recognised across scientific disciplines, using chemistry and computational linguistics as our model disciplines for these experiments.

The categories just introduced are abstract and depend on the annotators' interpretation of a rhetorical argument. This means that there is no guarantee that several independent annotators would annotate similarly. It is therefore crucial that all annotations at a high level of interpretation are backed up by human annotation with more than one annotator. However, annotations of citation function classification typically use only the untested annotation of a single human annotator as gold standard, who is typically the designer of a scheme (Spiegel-Rüsing, 1977; Weinstock, 1971; Nanba and Okumura, 1999; Garzone and Mercer, 2000). Teufel et al. (2006) are the only exception who test their citation function scheme using modern corpus-linguistic annotation methodology.

A study of human agreement on AZ annotation exists (Teufel et al., 1999), but this uses articles from only one discipline, namely computational linguistics. In this paper, we use a similar methodology to Teufel et al., but with data from two disciplines. The preliminary conclusion from these experiments is that annotation with chemistry papers has resulted in higher agreement than annotation with computational linguistics papers.

We extend the AZ annotation scheme to make further distinctions, as will be discussed in section 2. We also created an environment in which domain knowledge that an annotator might have about the science in a paper is systematically disregarded. We will describe how this was done in section 3, and then present the annotation experiment itself in section 4.

2 Changes to the AZ Scheme

Argumentative Zoning II (AZ-II) is a new annotation scheme, which is an elaboration of the original AZ scheme. It is presented in Fig. 2. Our annotation guidelines are 111 sides of A4 and contain a decision tree, detailed description of the semantics of the 15 categories, 75 rules for pairwise distinction of the categories and copious examples from both chemistry and computational linguistics. During guideline development, 70 chemistry papers and 20 CL papers were used, which are distinct from the ones used for annotation. It took 3 months part-time-work to prepare the guidelines for CL, and substantially less time to adapt them for chemistry. We have made them available at `www.cl.cam.ac.uk/research/nl/sciborg`.

The differences between the original AZ and AZ-II are as follows:

- Category AIM remained the same.
- Category BACKGROUND was renamed CO_GRO, or common ground.
- Category OTHER was split into other people's work (OTHR) and the authors' own previous work (PREV_OWN).
- Category BASIS was split into usage (USE) and support (SUPPORT).
- Category CONTRAST was split into neutral comparison (CODI), contradiction (ANTISUPP), and a category combining research gaps with criticism (GAP_WEAK).
- Category OWN was split into description of method (OWN_MTHD), results (OWN_RES) and conclusions (OWN_CONC), and a category which specifies recoverable errors made by the authors (OWN_FAIL).

Category	Description	Category	Description
AIM	Statement of specific research goal, or hypothesis of current paper	OWN_CONC	Findings, conclusions (non-measurable) of own work
NOV_ADV	Novelty or advantage of own approach	CODI	Comparison, contrast, difference to other solution (neutral)
CO_GRO	No knowledge claim is raised (or knowledge claim not significant for the paper)	GAP_WEAK	Lack of solution in field, problem with other solutions
OTHR	Knowledge claim (significant for paper) held by somebody else. Neutral description	ANTISUPP	Clash with somebody else's results or theory; superiority of own work
PREV_OWN	Knowledge claim (significant) held by authors in a previous paper. Neutral description.	SUPPORT	Other work supports current work or is supported by current work
OWN_MTHD	New Knowledge claim, own work: methods	USE	Other work is used in own work
OWN_FAIL	A solution/method/experiment in the paper that did not work	FUT	Statements/suggestions about future work (own or general)
OWN_RES	Measurable/objective outcome of own work		

Figure 2: AZ-II Annotation Scheme.

- Category TEXTUAL was discontinued, because it is less informative than the other categories.
- Two new categories were introduced, NOV_ADV (advantages of the new knowledge claim) and FUT (declaration of limitations or future work).

Our AZ-II categories are more fine-grained than the original AZ categories. The reasons for this are twofold: To bring AZ closer to contemporary citation function schemes, and to incorporate distinctions recently found useful by other researchers. For instance, Chichester et al. (2005) argue that ANTISUPP is particularly important. The finer grain in AZ-II has been accomplished purely by splitting existing AZ categories; hence, the coarser AZ categories are recoverable (with the exception of the TEXTUAL category). Annotation examples are given in the appendix.

As in AZ, citations are an important but not necessarily decisive cue for a sentence to belong to a particular zone. The guidelines mention citations as one factor in deciding whether a knowledge claim holds, and citations occur in several examples, so it is likely that the presence of citations would have influenced annotators in their decision.

Of the changes, the distinction which is likely to have the greatest impact on the annotation is the split of OWN according to the stage of the authors' problem solving process – into methods, results, conclusion or local failure. In most life sciences, descriptions of research as a problem solving process are a dominant phenomenon, whereby problem-solving descriptions can be of differing length and embeddedness. For instance, in synthetic chemistry, the starting compound for the main synthesis in the paper may first have to be synthesised itself (if it is not commercially available, for instance). In that case, arriving at the compound is an intermediate, smaller problem-solving process which enables the larger problem-solving process that represents the new KC.

The original AZ scheme didn't mark the distinction, possibly because it is not as easily observable in CL as it is in the life sciences, and because problem-solving stages were not part of the main analytic interest of AZ, which focused on how scientific argumentation is related to descriptions of own and other work. Also, neither of the traditional AZ applications (summarisation or citation indexing) had any direct use for the subdivided categories. But in the life sciences, there are applications which would make use of such a subdivision. For instance, in chemistry there is a niche for search applications which guide searchers directly to the method and/or result sections in papers. Specifically, the OWN_FAIL category is motivated by the failure–and–recovery search. In text, OWN_FAILmarks cases where the authors helpfully mention in passing steps which were found not to work during a long synthetic procedure (often the 'total synthesis' of a compound which is found in nature). Such cases happen frequently, and are generally followed by a 'recovery' statement which explains how the problem can be avoided. Another possible application that calls for a subdivision is Feltrim et al.'s

(2005) rhetorical writing system for novice writers. It trains novices in writing rhetorically well-formed abstracts and therefore must have a way of distinguishing, for instance, between methods and results.

Note that several of the applications based on AZ and AZ-II in general rely on the rare categories much more than they rely on the more frequent categories. OWN_FAIL is an example of a rare but important category, and so is AIM, which is central to summarisation applications. The comparative and contrastive categories CODI ANTISUPP and GAP_WEAK, on the other hand, are particularly useful to citation-based search applications.

Other AZ-like schemes for scientific discourse created for the biomedical domain (Mizuta and Collier, 2004) and for computer science (Feltrim et al., 2005) also made the decision to subdivide OWN, in similar ways to how we propose here. The current work, however, is the first experimental proof that humans can make this distinction – and the others encoded in AZ-II – reliably, and in two quite distinct disciplines.

3 Discipline-Independent Non-Expert Annotation

An important principle of AZ is that its categories can be decided without domain knowledge. This rule is anchored in the guidelines: when choosing a category, no reasoning about the scientific facts is allowed. The avoidance of domain-knowledge has its motivation in a strategy for a hypothetical automatic text-understanding system for unrestricted texts. Given the state of the art in text processing and knowledge representation, text understanding systems should in our opinion use general, rhetorical, and logical aspects of the text, rather than attempting to recognise or represent the scientific knowledge contained in the text. What the human annotation – the gold standard – should then do is to simulate the best possible output that such a system could theoretically create.

Annotators may use only general, rhetorical or linguistic knowledge; knowledge which is shared by all proficient speakers of a language. The guidelines spell out what is meant by these general principles. For instance, one can use lexical and syntactic parallelism in a text to infer that the authors were setting up a comparison between themselves and some other approach.

There is, however, a problem with annotator expertise and with the exact implementation of the "no domain knowledge" principle. This problem does not become apparent until one starts working with disciplines where at least some of the annotators or guideline developers are not domain experts (chemistry, in our case). Domain experts naturally use scientific knowledge and inference when they make annotation decisions. It would be unrealistic to expect them to be able to disregard their domain knowledge simply because they were *instructed* to do so. Additionally, when all annotators/scheme developers are domain experts, it is hard to even notice the cases where they "accidentally" use domain knowledge during annotation. We therefore artificially created a situation where all annotators are "semi-informed non-experts", which forces them to comply with the principle, namely by the following rules:

Justification: Annotators have to justify all annotation decisions by pointing to some text-based evidence, and by giving the section heading in the guidelines that describes the particular reason for assigning the category. General discipline-specific knowledge an annotator may happen to have is excluded as justification. Annotators' justifications have to be typed into the annotation tool and are open to challenge during the training phase. Much of the allowable justification comes in the form of general and linguistic principles, e.g., an explicit cue phrase, the title, or the structural similarity of textual strings. For instance, annotators are allowed to infer that process-VPs in the title are likely to be the contribution (knowledge of the actual concrete contribution of a paper is a requirement for annotation of AIM).

Discipline-specific Generics: The guidelines contain a section with high-level facts about the general research practices in the discipline. These generics constitute the only *scientific* knowledge which is acceptable as a justification, and are aimed to help non-expert annotators recognise how a paper might relate to already established scientific knowledge, so that they will be able to avoid common mistakes about the knowledge claim status of a certain fact. For instance, the better they are able to distinguish what is commonly known from what is newly claimed by the authors, the more consistent their annotation will be.

Annotation with expert-trained non-expert annotators means that a domain expert must be available initially, during the development of the anno-

tation scheme and the guidelines, either as a co-developer or as an informant. The domain expert's job is to describe scientific knowledge in that domain in a general way, in as far as it is necessary for the scheme's distinctions, and to write the domain-specific rules for the individual categories, including the choice of example sentences. This means that the guidelines are split into a domain-general and a domain-specific part.

The discipline-specific generics in chemistry come in the form of a "chemistry primer", a 10-page collection of high-level scientific domain knowledge. It contains: a glossary of words a non-chemist would not have heard about or would not necessarily recognise as chemical terminology; a list of possible types of experiments performed in chemistry; a list of commonly used machinery; a list of non-obvious negative characterisations of experiments and compounds ("sluggish", "inert"); and a list of possible types of knowledge claims. For instance, in chemistry each chemical substance mentioned can have in principle a knowledge claim associated with its discovery or invention – with the exception of water, rock salt, the metals known in prehistory and a few others. If a compound or process is however considered to be so commonly used that it is in the "general domain" (e.g., "the Stern–Volmer equation" or "the Grignard reaction"), it is no longer associated with somebody's knowledge claim, and as a result its usage is not to be marked with category USE.

Descriptions of individual categories can have domain-specific subsections, as well as the general ones. For instance, if the text states that the authors could not replicate a published result, the guidelines describe the cases when this is the authors' fault (OWN_FAIL) in contrast to the cases where this is an indirect accusation of the previous experiment (ANTISUPP).

Another potentially unclear distinction is between results (OWN_RES) and conclusions (OWN_CONC). The difference is defined on the basis of how much reasoning is necessary to be able to make the statement concerned. If all the authors did was to read a measurement off an instrument, the label OWN_RES applies. Reasoning points to OWN_CONC; it is sometimes linguistically marked ("therefore", "we conclude", "this means that"), but in many cases, domain knowledge may be required to decide whether reasoning was necessary to make a certain statement. Possible OWN_RES statements, according to the chemistry primer, include: statements of simple numerical result; descriptions of graphs; descriptions of atoms' positions in three-dimensional space; statements of trends, unless a reason for these results is given; comparisons of results of more than one experiment, unless a reason for these results is given.

The chemistry primer also lists phenomena which in a typical experiment would be read off chemical machinery (e.g., "Stark effect"). This list gives the non-expert annotator an objective criterion to answer the question how likely it is that a certain statement by the authors was arrived at by inference. We also found that our list of phenomena which can be read off machinery, which was compiled from the first 30 papers, generalised well to the other 40 papers considered.

The chemistry primer is not an attempt to summarise all methods and experimentation types in chemistry; this would be impossible to do, certainly in a few pages. Rather, it tries to answer many of the high-level questions a non-expert would have to an expert, in the framework of AZ.

This methodology allows to hire expert and non-expert annotators and bring them in line with each other. We believe it could be expanded relatively easily into many other disciplines, using domain experts which create similar primers for genetics, experimental physics, cell biology, but re-using the bulk of the guidelines.

4 Annotation Experiments

The annotators were the co-developers of the annotation scheme and the authors of this paper. Whereas all three annotators have good background knowledge in CL, the largest difference between them concerns their expertise in chemistry: Annotator A is a PhD-level chemist, Annotator B has two years' of undergraduate training in chemistry and can therefore be considered a chemical semi-expert, and Annotator C has no specialised chemistry knowledge.

As agreement measure we choose the Kappa coefficient κ (Fleiss, 1971; Siegel and Castellan, 1988), the agreement measure predominantly used in natural language processing research (Carletta, 1996). κ corrects raw agreement $P(A)$ for agreement by chance $P(E)$:

$$\kappa = \frac{P(A)-P(E)}{1-P(E)}$$

No matter how many items or annotators, or how the categories are distributed, $\kappa = 0$ when there is no agreement other than what would be expected by chance, and $\kappa = 1$ when agreement is perfect. If two annotators agree *less* than expected by chance, κ can also be negative. Chance agreement $P(E)$ is defined as the level of agreement which would be reached by random annotation using the same distribution of categories as the real annotators. All work done here is reported in terms of Fleiss' κ. [1] κ is also designed to abstract over the number of annotators as its formula relies on the proportion of expected vs. observed *pairwise* agreements possible in a pool. That is, κ for k annotators will be an average of the values of κ taking all possible m-tuples of annotators from the annotator pool (with $m < k$). As a side effect of its definition of random agreement, κ treats agreement in a rare category as more surprising, and rewards such agreement more than an agreement in a frequent category. This is a desirable property, because we are more interested in the performance of the rare rhetorical categories than we are in the performance of the more frequent zone categories.

4.1 Data

For chemistry, 30 random-sampled papers from journals published between 2004 and 2007 by the Royal Society of Chemistry were used for annotation[2]. The papers cover all areas of chemistry and some areas close to chemistry, such as climate modelling, process engineering, and a double-blind medical trial. The data used for the experiment contains a total of 3745 sentences.

For computational linguistics, 9 papers were annotated, with a total of 1629 sentences. The papers were published between 1998 and 2001 at ACL, EACL or EMNLP conferences, and were taken from the Computation and Language archive. Both chemistry and CL papers were automatically sentence-split, with manual correction of errors; acknowledgement sections were disregarded. A web-based annotation tool was used for guideline definition and for annotation.

Category	Chem	CL	Category	Chem	CL
OWN_MTHD	25.4	55.6	SUPPORT	1.5	0.7
OWN_RES	24.0	5.6	GAP_WEAK	1.1	1.0
OWN_CONC	15.1	10.7	FUT	1.0	1.4
OTHR	8.3	10.0	NOV_ADV	1.0	0.8
USE	7.9	2.7	CODI	0.8	1.2
CO_GRO	6.7	5.7	OWN_FAIL	0.8	0.1
PREV_OWN	3.4	1.7	ANTISUPP	0.5	0.6
AIM	2.3	1.8			

Figure 3: Frequency of AZ-II Categories (in %).

Our choice of which data sets to use was effected by the relative length of papers more than by the journal/conference distinction. Average article length between chemistry journal articles (3650 words/paper) and CL conference articles (4219 words/paper) is comparable, so conference articles in CL seem a much better choice for comparative work than journal publications, which are often very long in CL. Additionally, conferences have a high profile in CL, and we found the conference publications to be of high editorial quality. We are nevertheless interested in the structure of longer journal articles, and plan to investigate CL journals in the future.

The annotations were done using a web-based annotation tool. Every sentence is assigned a category. No communication between the annotators was allowed.

4.2 Results

The inter-annotator agreement for chemistry was κ = 0.71 (N=3745,n=15,k=3). For CL, the inter-annotator agreement was κ = 0.65 (N=1629,n=15,k=3). For comparison, the inter-annotator agreement for the original, CL-specific AZ with 7 categories was κ = 0.71 (N=3420,n=7,k=3). Given the subjective nature of the task and the fact that AZ-II introduces additional distinctions, the AZ-II agreement can be considered acceptable for CL and relatively high for chemistry. Additionally, chemistry annotation used one non-expert annotator, who had no chemistry-specific domain knowledge apart from that in the chemistry primer.

The distribution of categories for the two disciplines is given in Fig. 3. As expected, there is a large discrepancy in frequency between the (rare) rhetorical categories and the (much more frequent) zone categories OWN_MTHD, OWN_RES,

[1] Artstein and Poesio (2008) observe that there are several version of κ which differ in how $P(E)$ is calculated. In particular, Fleiss' (1971) κ calculates $P(E)$ as the average observed distribution of all annotators, whereas Cohen's (1960) κ calculates $P(E)$ only on the basis of the other annotator(s).

[2] 100 papers across a spread of disciplines from the January 2004 issues of the RSC were selected blindly (but with an attempt to cover most areas of chemistry). 30 out of these were random sampled for annotation; the rest were used for annotation development.

OWN_CONC, OTHR and CO_GRO. For supervised learning, too few examples of any category can be a problem. There are methods which attempt to reduce the annotation effort by using a trained classifier to suggest possible cases to a human. However, the classifier can only find examples similar to the ones that have already been manually classified, when the real problem is a recall-problem, i.e., the challenge is to find more new examples in the multitude of possible sentences. To solve this in a fundamentally sound way, there seems to be no other way than to annotate more texts, at the cost of more human effort.

If we consider the differences across disciplines, the most striking ones concern the distribution of OWN_MTHD, which is more than twice as common in CL (56% v. 25%), and OWN_RES, which is far more common in chemistry overall (24% v 5.6%). Usage of other people's knowledge claims or materials also seems to be more common in chemistry, or at least more explicitly expressed (7.9% vs 2.7%). With respect to the shorter, rarer categories, there is a marked difference in OWN_FAIL (0.1% in CL, but 0.8% in chemistry[3] and SUPPORT, which is more common in chemistry (1.5% vs 0.7%). However, this effect is not present for ANTISUPP (contradiction of results), the "reverse" category to SUPPORT, (0.6% in CL vs 0.5% in chemistry).

As far as the chemistry annotation is concerned, it is interesting to find out whether Annotator A was influenced during annotation by domain knowledge which Annotator C did not have, and Annotator B had to a lower degree[4]. We therefore calculated pairwise agreement, which was $\kappa_{\mathrm{AC}} = 0.66$, $\kappa_{\mathrm{BC}} = 0.73$ and $\kappa_{\mathrm{AB}} = 0.73$ (all: N=3745,n=15,k=2). That means that the largest disagreements were between the non-expert (C) and the expert (A), though the differences are modest. This might point to the fact that Annotators A and B might have used a certain amount of domain-knowledge which the chemistry primer in the guidelines does not yet, but should, cover.

In an attempt to determine how well categories are defined, we first consider the binary distinction between zone categories (OWN_MTHD, OWN_RES, OWN_CONC, OWN_FAIL, OTHR, PREV_OWN and CO_GRO) and rhetorical categories (the other 8). This shows an inter-annotator agreement of κ_{binary} = 0.78 (N=3745, n=2, k=3) for chemistry and κ_{binary} = 0.65 (N=1629, n=2, k=3) for CL, indicating that annotators find it relatively easy (chemistry) or at least not more difficult than the overall distinction (CL) to distinguish these two types of categories. We next perform Krippendorff's (1980) category distinctions (Fig. 4). Here, all categories apart from the one diagnosed are collapsed, and what is reported is the difference of inter-annotator agreement when compared to the overall distinctiveness (κ=0.71 for chemistry, κ=0.65 for CL). Where the difference is positive, the annotators could distinguish the given category better than they could distinguish all categories, and where they are negative, correspondingly worse.[5]

The results confirm that categories USE, AIM, OWN_MTHD, OWN_RES and FUT are particularly well distinguished in both disciplines. This is a positive result, as these categories are important for several types of searches. In these cases the guidelines seem to fully suffice for their description, but then again good performance of AIM, FUT and USE is not that surprising, as they are signalled clearly by linguistic and non-linguistic cues. However, there are three categories with particularly low distinguishability in both disciplines: ANTISUPP, OWN_FAIL and PREV_OWN. As ANTISUPP and OWN_FAIL are crucial for the envisaged downstream tasks, the problems with their definition should be identified and fixed. We are in the process of systematically troubleshooting the guidelines for those categories.

The table also shows that category definition has discipline-specific problems. For instance, we believe that the fact that distinctiveness for OWN_FAIL is so bad for CL must be due to the fact that we only encountered very few potential OWN_FAIL cases in this domain. The definition of the categories SUPPORT and NOV_ADV also seem to be substantially more confusing for CL than for chemistry. However, CODI is a category which shows average distinctiveness for CL, but much worse distinctiveness for chemistry. We believe this is due to the fact that comparisons of

[3]These are not large differences in absolute terms – 55 items identified as OWN_FAIL by at least one annotator in chemistry, vs. 7 such items in CL, the relative difference is large and confirms that in chemistry papers, particularly descriptions of synthesis procedures, OWN_FAIL cases appear relatively frequently.

[4]This question does not arise in the case of CL, as all annotators can be considered experts in this respect.

[5]All κ values for chemistry were measured with N=3745, n=2, k=3; for CL with N=1629, n=2, k=3.

methods and approaches are more common in CL and are clearly expressed, whereas in chemistry the objects that are involved in comparisons are more varied and at a lower grade of abstraction (e.g., compounds, properties of compounds, coefficients, etc.), which obviously has a negative effect on the distinctiveness of this category.

Category	Chem	CL	**Category**	Chem	CL
USE	+0.12	+0.00	NOV_ADV	-0.07	-0.23
AIM	+0.09	+0.08	OWN_CONC	-0.08	-0.13
OWN_MTHD	+0.05	+0.05	GAP_WEAK	-0.08	-0.16
OWN_RES	+0.02	+0.04	PREV_OWN	-0.11	-0.15
FUT	+0.01	+0.06	OWN_FAIL	-0.19	-0.43
CO_GRO	-0.01	-0.03	ANTISUPP	-0.35	-0.32
SUPPORT	-0.04	-0.12	CODI	-0.36	+0.00
OTHR	-0.06	+0.07			

Figure 4: Krippendorff's Diagnostics for Category Distinction (κ, relative to Overall Distinctiveness).

We also provide a direct comparison of our annotation results with those from the original AZ scheme. Comparisons between two similar annotation schemes can be made by collapsing those categories in each scheme which are not distinguished in the other scheme. Such a comparison can of course only ever approximate the smallest common denominator between two schemes.

The AZ-II categories were collapsed into a set of six categories that closely resemble AZ categories, as described in section 2 (with OWN simulated by the union of OWN_FAIL, OWN_MTHD, OWN_RES, OWN_CONC, FUT, and NOV_ADV). This created a 6-category AZ annotation.

As TEXTUAL is not marked up in AZ-II, the original AZ annotation was also collapsed, by incorporating TEXTUAL examples into OWN. The two 6-pronged AZ-annotations are now more directly comparable. Inter-annotator agreement for the collapsed AZ-II showed κ = 0.75 (N=3745, n=6, k=3). This compares favourably to the collapsed AZ's agreement of κ = 0.71 (N=3420, n=6, k=3); but when comparing the raw numerical results one should consider that different data from different disciplines is used (chemistry in AZ-II, CL in AZ).

These results should be interpreted as a positive result for the domain-independence of AZ, and also for the feasibility of using trained non-experts as annotators. The additional work that went into the guidelines has produced annotation of a high consistency, even though AZ-II provides more distinctions (15 categories vs. 7 in AZ).

There is also the faint possibility that discourse annotation of chemistry is intrinsically easier than discourse annotation of CL, *because* it is a more established discipline and not despite of it. For instance, it is likely that the problem-solving categories OWN_FAIL, OWN_MTHD, OWN_RES and OWN_CONC are easier to describe in a discipline with an established methodology (such as chemistry), than they are in a younger, developing discipline such as computational linguistics.

5 Conclusion

Argumentative Zoning is an analysis of the rhetorical progression of the scientific argument in a paper. In this paper, we have made the following contributions to this analysis:

- We have presented a more informative scheme, which additionally recognises the structure of an experiment in terms of problem solving (method – results – conclusions) and makes more fine-grained distinctions in some of the sentiment-inspired relational categories (e.g., criticism and comparisons to other approaches).
- We introduced an annotation methodology which attempts to systematically exclude the use of annotators' extraneous domain knowledge from the annotation.
- We have experimentally shown that human coders can independently annotate this new AZ scheme in two distinct disciplines. Our results show inter-annotator agreements of κ=0.65 and κ=0.71 for computational linguistics and chemistry, respectively.

Overall, the outcome of this work indicates that the phenomena described in AZ can be defined in a domain-independent way. The experiment also tested how realistic the "expert-trained non-expert" approach to domain-knowledge free annotation is. The fact that the agreement between three annotators (an expert, a semi-expert, and a non-expert) is acceptable overall vindicate our task definition as domain-knowledge free (using the tools of justification and domain-specific generic knowledge). However, the agreements involving the semi-expert are higher than the agreement between expert and non-expert. This probably means that the chemistry generics were not fully adequate to ensure that the non-expert understood enough of the chemistry to achieve the highest-possible agreement.

The automation of AZ-annotation is underway. This requires adaptation of the high-level features used in AZ (Teufel and Moens, 2002) to chemistry. We are also preparing an annotation experiment with naive annotators. Another research avenue is the expansion of the guidelines to other disciplines such as bio-medicine, and to longer journal articles, e.g., in computational linguistics.

6 Acknowledgements

This work was funded by EPSRC project Sciborg (EP/C010035/1).

Appendix: Annotation Examples[6]

AIM *We now describe in this paper a synthetic route for the functionalisation of the framework of mesoporous organosilica by free phosphine oxide ligands, which can act as a template for the introduction of lanthanide ions.* (b514878b)

AIM *The aim of this paper is to examine the role that training plays in the tagging process ...* (9410012)

NOV_ADV *Moreover, the simplicity and ease of application of the electrochemical method [...] should also be emphasised and makes it an interesting and valuable synthetic tool.* (b513402a)

NOV_ADV *Other than the economic factor, an important advantage of combining morphological analysis and error detection/correction is the way the lexical tree associated with the analysis can be used to determine correction possibilities.* (9504024)

CO_GRO *A wide range of organosulfur compounds are biologically active and some find commercial application as fungicides and bactericides*$^{1-4}$. (b514441h)

CO_GRO *It has often been stated that discourse is an inherently collaborative process ...* (9504007)

OTHR *In their system, antibody immobilized on a solid substrate reacts with antigen, which binds with another antibody labelled with peroxidase.* (b313094k)

OTHR *But in Moortgat's mixed system all the different resource management modes of the different systems are left intact in the combination and can be exploited in different parts of the grammar.* (9605016)

PREV_OWN *As a program aimed at the applications of imines*$^{(2a,g,5)}$ *we have studied the formation of carbanions from imines and their subsequent reactions.* (b200198e)

PREV_OWN *Earlier work of the author (Feldweg 1993; Feldweg 1995a) within the framework of a project on corpus based development of lexical knowledge bases (ELWIS) has produced LIKELY ...* (9502038)

OWN_MTHD *In order for it to be useful for our purposes, the following extensions must be made:* (0102021)

OWN_MTHD *On the other hand, a tertiary amide can be an excellent linking functional group.* (b201987f)

OWN_FAIL *Initial attempts to improve the dehydration of* **4** *via chemical or thermal means were unsuccessful; similarly, attempts to couple the chlorosilane (Me3Si)2 (Me2ClSi)CH with Ag2O failed.* (b510692c)

OWN_FAIL *When the ABL algorithms try to learn with two completely distinct sentences, nothing can be learned.* (0104006)

OWN_RES *While the acid 1a readily coupled to the olefin, the corresponding boronic ester was surprisingly inert under the reaction conditions.* (b311492a)

OWN_RES *All the curves have a generally upward trend but always lie far below backoff (51% error rate).* (0001012)

OWN_CONC *It is unlikely that every VOC emit ted by plants serves an ecological or physiological role ...* (b507589k)

OWN_CONC *Unless grammar size takes on proportionately much more significance for such longer inputs, which seems implausible, it appears that in fact the major problems do not lie in the area of grammar size, but in input length.* (9405033)

GAP_WEAK *Various methods of preparation have been developed, but they often suffer from low yield and tedious separation.*[16,17,28,31] (b200888m)

GAP_WEAK *Here, we will produce experimental evidence suggesting that this simple model leads to serious overestimates of system error rates...* (9407009)

CODI *However, the measured values of the dielectric constant (ϵ = 310) are lower than the values reported by Ganguli and coworkers*$^{(21)}$ *for BSTO pellets sintered at 1100* $\mathrm{deg}C$ *...* (b506578j)

CODI *Unlike most research in pragmatics that focuses on certain types of presuppositions or implicatures, we provide a global framework in which one can express all these types of pragmatic inferences.* (9504017)

SUPPORT *This is in line with the findings of Martin and Illas for inorganic solids* $^{(84,85)}$. (b515732c)

SUPPORT *Work similar to that described here has been carried out by Merialdo (1994), with broadly similar conclusions.* (9410012)

USE *The diamine* **10** *was prepared following a previously published procedure*$^{(4d)}$. (b110865b)

USE *We use the framework for the allocation and transfer of control of Whittaker and Stenton (1988).* (9504007)

FUT *Our further efforts are directed towards the above goal,... and overcoming limitations pertaining to the electron-poor arylboronic acids.* (b311492a)

FUT *An important area for future research is to develop principled methods for identifying distinct speaker strategies pertaining to how they signal segments.* (9505025)

ANTISUPP *Although purification of 8b to a de of 95percent has been reported elsewhere*[31]*, in our hands it was always obtained as a mixture of the two [EQN]-diastereomers.* (b310767a)

ANTISUPP *This result challenges the claims of recent discourse theories (Grosz and Sidner 1986, Reichman 1985) which argue for a the close relation between cue words and discourse structure.* (9504006)

[6] Corpus examples are taken from our chemistry and CL data sets; indicated by their respective file numbers.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Jean Carletta. 1996. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254.

Christine Chichester, Frdrique Lisacek, Aaron Kaplan, and Agnes Sandor. 2005. Discovering paradigm shift patterns in biomedical abstracts: application to neurodegenerative diseases. In *Proceedings of First International Symposium on Semantic Mining in Biomedicine*.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46.

V. Feltrim, Simone Teufel, Gracas Nunes, and S. Alusio. 2005. Argumentative zoning applied to critiquing novices' scientific abstracts. In Janyce Wiebe James G. Shanahan, Yan Qu, editor, *Computing Attitude and Affect in Text: Theory and Applications*, pages 233–245. Springer, Dordrecht, The Netherlands.

J. L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76:378–381.

Eugene Garfield. 1965. Can citation indexing be automated? In M. et al. Stevens, editor, *Statistical Association Methods for Mechanical Documentation (NBS Misc. Pub. 269)*. National Bureau of Standards, Washington.

Mark Garzone and Robert E. Mercer. 2000. Towards an automated citation classifier. In *Proceedings of the 13th Biennial Conference of the CSCI/SCEIO (AI-2000)*, pages 337–346.

Ken Hyland. 1998. Persuasion and context: The pragmatics of academic metadiscourse. *Journal of Pragmatics*, 30(4):437–455.

Klaus Krippendorff. 1980. *Content Analysis: An Introduction to its Methodology*. Sage Publications, Beverly Hills, CA.

Terttu Luukkonen. 1992. Is scientists' publishing behaviour reward-seeking? *Scientometrics*, 24:297–319.

Yoko Mizuta and Nigel Collier. 2004. An annotation scheme for rhetorical analysis of biology articles. In *Proceedings of LREC'2004*.

Greg Myers. 1992. In this paper we report...—speech acts and scientific facts. *Journal of Pragmatics*, 17(4):295–313.

Hidetsugu Nanba and Manabu Okumura. 1999. Towards multi-paper summarization using reference information. In *Proceedings of the XXth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 926–931.

Sidney Siegel and N. John Jr. Castellan. 1988. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, Berkeley, CA, 2nd edition.

Ina Spiegel-Rüsing. 1977. Bibliometric and content analysis. *Social Studies of Science*, 7:97–113.

John Swales, 1990. *Genre Analysis: English in Academic and Research Settings. Chapter 7: Research articles in English*, pages 110–176. Cambridge University Press, Cambridge, UK.

Simone Teufel and Marc Moens. 2002. Summarising scientific articles — experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–446.

Simone Teufel, Jean Carletta, and Marc Moens. 1999. An annotation scheme for discourse-level argumentation in research articles. In *Proceedings of the 9th Meeting of the European Chapter of the Association for Computational Linguistics (EACL-99)*, pages 110–117, Bergen, Norway.

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *Proceedings of EMNLP-06*.

Simone Teufel. 2001. Task-based evaluation of summary quality: Describing relationships between scientific papers. In *Proceedings of NAACL-01 Workshop "Automatic Text Summarization"*, Pittsburgh, PA.

Melvin Weinstock. 1971. Citation indexes. In *Encyclopedia of Library and Information Science*, volume 5, pages 16–40. Dekker, New York, NY.

Character-level Analysis of Semi-Structured Documents for Set Expansion

Richard C. Wang
Language Technologies Institute
Carnegie Mellon University
rcwang@cs.cmu.edu

William W. Cohen
Machine Learning Department
Carnegie Mellon University
wcohen@cs.cmu.edu

Abstract

Set expansion refers to expanding a partial set of "seed" objects into a more complete set. One system that does set expansion is SEAL (Set Expander for Any Language), which expands entities automatically by utilizing resources from the Web in a language-independent fashion. In this paper, we illustrated in detail the construction of character-level wrappers for set expansion implemented in SEAL. We also evaluated several kinds of wrappers for set expansion and showed that character-based wrappers perform better than HTML-based wrappers. In addition, we demonstrated a technique that extends SEAL to learn binary relational concepts (e.g., "x is the mayor of the city y") from only two seeds. We also show that the extended SEAL has good performance on our evaluation datasets, which includes English and Chinese, thus demonstrating language-independence.

1 Introduction

SEAL[1] (Set Expander for Any Language) is a set expansions system that accepts input elements (seeds) of some target set S and automatically finds other probable elements of S in semi-structured documents such as web pages. SEAL is a research system that has shown good performance in previously published results (Wang and Cohen, 2007). By using only three seeds and top one hundred documents returned by Google, SEAL achieved 90% in mean average precision (MAP), averaged over 36 datasets from three languages: English, Chinese, and Japanese. Unlike other published research work (Etzioni et al., 2005), SEAL focuses on finding small closed sets of items (e.g., Disney movies) rather than large and more open sets (e.g., scientists).

In this paper, we explore the impact on performance of one of the innovations in SEAL, specifically, the use of character-level techniques to detect candidate regular structures, or *wrappers*, in web pages. Although some early systems for web-page analysis induce rules at character-level (e.g., such as WIEN (Kushmerick et al., 1997) and DIPRE (Brin, 1998)), most recent approaches for set expansion have used either tokenized and/or parsed free-text (Carlson et al., 2009; Talukdar et al., 2006; Snow et al., 2006; Pantel and Pennacchiotti, 2006), or have incorporated heuristics for exploiting HTML structures that are likely to encode lists and tables (Nadeau et al., 2006; Etzioni et al., 2005).

In this paper, we experimentally evaluate SEAL's performance under two settings: 1) using the character-level page analysis techniques of the original SEAL, and 2) using page analysis techniques constrained to identify only HTML-related wrappers. Our conjecture is that the less constrained character-level methods will produce more candidate wrappers than HTML-based techniques. We also conjecture that a larger number of candidate wrappers will lead to better performance overall, due to SEAL's robust methods for ranking candidate wrappers.

The experiments in this paper largely validate this conjecture. We show that the HTML-restricted version of SEAL performs less well, losing 13 points in MAP on a dozen Chinese-language benchmark problems, 8 points in MAP on a dozen English-language problems, and 2 points in MAP on a dozen Japanese-language problems.

SEAL currently only handles unary relationships (e.g., "x" is a mayor). In this paper, we show that SEAL's character-level analysis techniques can, like HTML-based methods, be read-

[1] http://rcwang.com/seal

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1503–1512,
Singapore, 6-7 August 2009.

ily extended to handle binary relationships. We then demonstrate that this extension of SEAL can learn binary concepts (e.g., "x is the mayor of the city y") from a small number of seeds, and show that, as with unary relationships, MAP performance is 26 points lower when wrappers are restricted to be HTML-related. Furthermore, we also illustrate that the learning of binary concepts can be bootstrapped to improve its performance.

Section 2.1 explains how SEAL constructs wrappers and rank candidate items for unary relations. Section 3 describes the experiments and results for unary relations. Section 4 presents the method for extending SEAL to handle binary relationships, as well as their experimental results. Related work is discussed in Section 5, and the paper concludes in Section 6.

2 SEAL

2.1 Identifying Wrappers for Unary Relations

When SEAL performs set expansion, it accepts a small number of *seeds* from the user (e.g., "Ford", "Nissan", and "Toyota"). It then uses a web search engine to retrieve some documents that contain these instances, and then analyzes these documents to find *candidate wrappers* (i.e., regular structures on a page that contain the seed instances). Strings that are extracted by a candidate wrapper (but are not equivalent to any seed) are called *candidate instances*. SEAL then statistically ranks the candidate instances (and wrappers), using the techniques outlined below, and outputs a ranked list of instances to the user.

One key step in this process is identifying candidate wrappers. In SEAL, a candidate wrapper is defined by a pair of left and right character strings, ℓ and r. A wrapper "extracts" items from a particular document by locating all strings in the document that are bracketed by the wrapper's left and right strings, but do not contain either of the two strings. In SEAL, wrappers are always learned from, and applied to, a single document.

Table 1 illustrates some candidate wrappers learned by SEAL. (Here, a wrapper is written as ℓ`[...]`r, with the `[...]` to be filled by an extracted string.) Notice that the instances extracted by wrappers can and do appear in surprising places, such as embedded in URLs or in HTML tag attributes. Our experience with these character-based wrappers lead us to conjecture that existing heuristics for identifying structure in HTML are fundamentally limited, in that many potentially useful structures will not be identified by analyzing HTML structure only.

SEAL uses these rules to find wrappers. Each candidate wrapper ℓ, r is a maximally long pair of strings that bracket at least one occurrence of every seed in a document: in other words, for each pair ℓ, r, the set of strings C extracted by ℓ, r has the properties that:

1. For every seed s, there exists some $c \in C$ that is equivalent to s; and
2. There are no strings ℓ', r' that satisfy property (1) above such that ℓ is a proper suffix of ℓ' and r is a proper prefix of r'.

SEAL's wrappers can be found quite efficiently. The algorithm we use has been described previously (Wang and Cohen, 2007), but will be explained again here for completeness. As an example, below shows a mock document, written in an unknown mark-up language, that has the seeds: Ford, Nissan, and Toyota located (and boldfaced). There are two other car makers hidden inside this document (can you spot them?). In this section, we will show you how to automatically construct wrappers that reveal them.

```
GtpKxH<b>nIsSaN</b>xjHJglekuDialcLBxKH<b>forD</b>xkrpW
NaCMwAAHO<b>FoRd</b>uohdEXocUvaGKxHaCuRAxjHjnOx
o<b>TOyOTa</b>zxKHAUdIxkrOyQKxH<b>ToYotA</b>xjHCRdmLxa
puRAPprtqOVKxH<b>foRd</b>xjHaJAScRFrla<b>FoRD</b>ofwNL
WxKH<b>tOYota</b>xkrHxQKlacXlGEKtxKH<b>NisSan</b>xkrEq
```

Given a set of seeds and a semi-structured document, the wrapper construction algorithm starts by locating all strings equivalent to a seed in the document; these strings are called *seed instances* below. (In SEAL, we always use case-insensitive string matching, so a string is "equivalent to" any case variant of itself.) The algorithm then inserts all the instances into a list and assigns a unique id to each of them by its index in the list (i.e., the id of an instance is its position in the list.)

For every seed instance in the document, its immediate left character string (starting from the first character of the document) and right character string (ending at the last character of the document) are extracted and inserted into a *left-context* trie and a *right-context* trie respectively, where the left context is inserted in reversed character order. (Here, we implemented a compact trie called

URL:	`http://www.shopcarparts.com/`
Wrapper:	`.html" CLASS="shopcp">[...] Parts</A>  `
Content:	acura, audi, bmw, buick, cadillac, chevrolet, chevy, chrysler, daewoo, daihatsu, dodge, eagle, ford, ...
URL:	`http://www.allautoreviews.com/`
Wrapper:	`</a>  <a href="auto_reviews/[...]/`
Content:	acura, audi, bmw, buick, cadillac, chevrolet, chrysler, dodge, ford, gmc, honda, hyundai, infiniti, isuzu, ...
URL:	`http://www.hertrichs.com/`
Wrapper:	`<li class="franchise [...]"> <h4><a href="#">`
Content:	buick, chevrolet, chrysler, dodge, ford, gmc, isuzu, jeep, lincoln, mazda, mercury, nissan, pontiac, scion, ...
URL:	`http://www.metacafe.com/watch/1872759/2009_nissan_maxima_performance/`
Wrapper:	`videos">[...]</a> <a href="/tags/`
Content:	avalon, cars, carscom, driving, ford, maxima, nissan, performance, speed, toyota
URL:	`http://www.worldstyling.com/`
Wrapper:	`'>[...] Accessories</option><option value='`
Content:	chevy, ford, isuzu, mitsubishi, nissan, pickup, stainless steel, suv, toyota

Table 1: Examples of wrappers constructed from web pages given the seeds: Ford, Nissan, Toyota.

Patricia trie where every node stores a substring.) Every node in the left-context trie maintains a list of ids for keeping track of the seed instances that follow the string associated with that node. Same thing applies to the right-context trie symmetrically. Figure 1 shows the two context tries and the list of seed instances when provided the mock document with the seeds: Ford, Nissan, and Toyota.

Provided that the left and right context tries are populated with all the contextual strings of every seed instance, the algorithm then finds maximally long contextual strings that bracket at least one seed instance of every seed. The pseudo-code for finding these strings for building wrappers is illustrated in Table 2, where $Seeds$ is the set of input seeds and ℓ is the minimum length of the strings. We observed that longer strings produce higher precision but lower recall. This is an interesting parameter that is worth exploring, but for this paper, we consider and use only a minimum length of one throughout the experiments. The basic idea behind the pseudo-code is to first find all the longest possible strings from one trie given some constraints, then for every such string s, find the longest possible string s' from another trie such that s and s' bracket at least one occurrence of every given seed in a document.

The wrappers constructed as well as the items extracted given the mock document and the example seeds are shown below. Notice that Audi and Acura are uncovered (did you spot them?).

Wrapper:	`xKH[...]xkr`
Content:	**audi**, ford, nissan, toyota
Wrapper:	`KxH[...]xjH`
Content:	**acura**, ford, nissan, toyota

Wrappers MakeWrappers(Trie ℓ, Trie r)
Return Wraps(l, r) $\cup$ Wraps(r, l)

Wrappers Wraps(Trie t_1, Trie t_2)
For each $n_1 \in$ TopNodes(t_1, ℓ)
For each $n_2 \in$ BottomNodes(t_2, n_1)
For each $n_1 \in$ BottomNodes(t_1, n_2)
Construct a new Wrapper(Text(n_1), Text(n_2))
Return a union of all wrappers constructed

Nodes BottomNodes(Trie t_1, Node n')
Find node $n \in t_1$ such that:
(1) NumCommonSeeds(n, n') == $|Seeds|$, and
(2) All children nodes of n (if exist) fail on (1)
Return a union of all nodes found

Nodes TopNodes(Trie t, int ℓ)
Find node $n \in t$ such that:
(1) Text(n).$length \geq \ell$, and
(2) Parent node of n (if exist) fails on (1)
Return a union of all nodes found

String Text(Node n)
Return the textual string represented by the path from root to n in the trie containing n

---

Integer NumCommonSeeds(Node n_1, Node n_2)
For each index $i \in$ Intersect(n_1, n_2):
Find the seed at index i of seed instance list
Return the size of the union of all seeds found

Integers Intersect(Node n_1, Node n_2)
Return $n_1.indexes \cap n_2.indexes$

Table 2: Pseudo-code for constructing wrappers.

Table 1 shows examples of wrappers constructed from real web documents. We have also observed items extracted from plain text (.txt), comma/tab-separated text (.csv/.tsv), latex (.tex), and even Word documents (.doc) of which the wrappers have binary character strings. These observations support our claim that the algorithm is independent of mark-up language. In our experimental results, we will show that it is independent of human language as well.

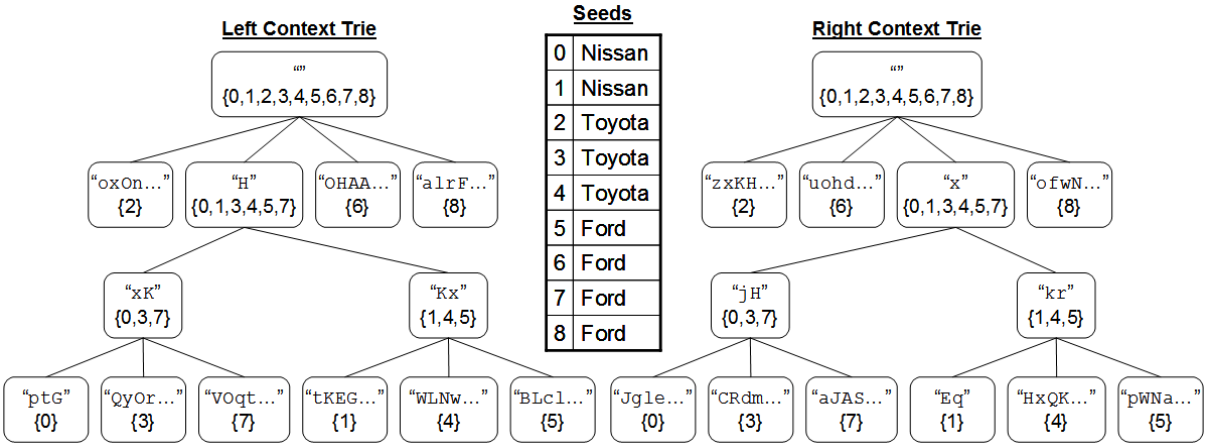


Figure 1: The context tries and the seed instance list constructed given the mock document presented in Section 2.1 and the seeds: Ford, Nissan and Toyota.

2.2 Ranking Wrappers and Candidate Instances

In previous work (Wang and Cohen, 2007), we presented a graph-walk based technique that is effective for ranking sets and wrappers. This model encapsulates the relations between documents, wrappers, and extracted instances (entity mentions). Similarly, our graph also consists of a set of nodes and a set of labeled directed edges. Figure 2 shows an example graph where each node d_i represents a document, w_i a wrapper, and m_i an extracted entity mention. A directed edge connects a node d_i to a w_i if d_i contains w_i, a w_i to a m_i if w_i extracts m_i, and a d_i to a m_i if d_i contains m_i. Although not shown in the figure, every edge from node x to y actually has an inverse relation edge from node y to x (e.g., m_i is extracted by w_i) to ensure that the graph is cyclic.

We will use letters such as x, y, and z to denote nodes, and $x \xrightarrow{r} y$ to denote an edge from x to y with labeled relation r. Each node represents an object (document, wrapper, or mention), and each edge $x \xrightarrow{r} y$ asserts that a binary relation $r(x, y)$ holds. We want to find entity mention nodes that are *similar* to the seed nodes. We define the similarity between two nodes by random walk with restart (Tong et al., 2006). In this algorithm, to walk away from a source node x, one first chooses an edge relation r; then given r, one picks a target node y such that $x \xrightarrow{r} y$. When given a source node x, we assume that the probability of picking an edge relation r is uniformly distributed among the set of all r, where there exist a target node y such that $x \xrightarrow{r} y$. More specifically,

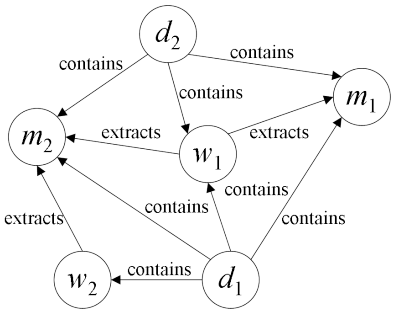


Figure 2: Example graph built by Random Walk.

$$P(r|x) = \frac{1}{|r : \exists y \; x \xrightarrow{r} y|} \tag{1}$$

We also assume that once an edge relation r is chosen, a target node y is picked uniformly from the set of all y such that $x \xrightarrow{r} y$. More specifically,

$$P(y|r, x) = \frac{1}{|y : x \xrightarrow{r} y|} \tag{2}$$

In order to perform random walk, we will build a transition matrix M where each entry at (x, y) represents the probability of traveling one step from a source node x to a target node y, or more specifically,

$$M_{xy} = \sum_r P(r|x)P(y|r, x) \tag{3}$$

We will also define a state vector $\vec{v}_t$ which represents the probability at each node after iterating through the entire graph t times, where one iteration means to walk one step away from every node. The state vector at $t + 1$ iteration is defined as:

$$\vec{v}_{t+1} = \lambda \vec{v}_0 + (1 - \lambda) M \vec{v}_t \tag{4}$$

Since we want to start our walk from the seeds, we initialize v_0 to have probabilities uniformly distributed over the seed nodes. In each step of our walk, there is a small probability λ of teleporting back to the seed nodes, which prevents us from walking too far away from the seeds. We iterate our graph until the state vector converges, and rank the extracted mentions by their probabilities in the final state vector. We use a constant λ of 0.01 in the experiments below.

2.3 Bootstrapping Candidate Instances

Bootstrapping refers to iterative unsupervised set expansion. This process requires minimal supervision, but is very sensitive to the system's performance because errors can easily propagate from one iteration to another. As shown in previous work (Wang and Cohen, 2008), carefully designed seeding strategies can minimize the propagated errors. Below, we show the pseudo-code for our bootstrapping strategy.

$stats \leftarrow \emptyset, used \leftarrow inputs$
for i = 1 to M **do**
  $m = min(3, |used|)$
  $seeds \leftarrow select_m(used) \cup top(list)$
  $stats \leftarrow expand(seeds, stats)$
  $list \leftarrow rank(stats)$
  $used \leftarrow used \cup seeds$
end for

where M is the total number of iterations, $inputs$ are the two initial input seeds, $select_m(S)$ randomly selects m different seeds from the set S, $used$ is a set that contains previously expanded seeds, $top(list)$ returns an item that has the highest rank in $list$, $expand(seeds, stats)$ expands the selected $seeds$ using $stats$ and outputs accumulated statistics, and $rank(stats)$ applies Random Walk described in Section 2.2 on the accumulated $stats$ to produce a $list$ of items. This strategy dumps the highest-ranked item into the $used$ bucket after every iteration. It starts by expanding two input seeds. For the second iteration, it expands three seeds: two *used* plus one from last iteration. For every successive iteration, it expands four seeds: three randomly selected $used$ ones plus one from last iteration.

3 Experiments with Unary Relations

We would like to determine whether character-based or HTML-based wrappers are more suited for the task of set expansion. In order to do that, we introduce five types of wrappers, as illustrated in Table 3. The first type is the character-based wrapper that does not have any restriction on the alphabets of its characters. Starting from the second type, the allowable alphabets in a wrapper become more restrictive. The fifth type requires that an item must be tightly bracketed by two complete HTML tags in order to be extracted.

#	L. Context `[...]` R. Context	Eng	Jap	Chi	Avg
1	`.+[...].+`	87.6	96.9	95.4	93.3
2	`.*[<>].*[...].*[<>].*`	85.7	96.8	90.7	91.1
3	`.*>[...]<.*`	85.7	96.7	90.7	91.0
4	`.*<.+?>.*[...].*<.+?>.*`	80.1	95.8	83.7	86.5
5	`.*<.+?>[...]<.+?>.*`	79.6	94.9	82.4	85.6

Table 3: The performance (MAP) of various types of wrappers on semi-structured web pages.

All pure HTML-based wrappers are type 5, possibly with additional restrictions imposed (Nadeau et al., 2006; Etzioni et al., 2005). SEAL currently does not use an HTML parser (or any other kinds of parser), so restrictions cannot be easily imposed. As far as we know, there isn't an agreement on what restrictions make the most sense or work the best. Therefore, we evaluate performance for varying wrapper constraints from type 1 (most general) to type 5 (most strict) in our experiments.

For set expansion, we use the same evaluation set as in (Wang and Cohen, 2007) which contains 36 manually constructed lists across three different languages: English, Chinese, and Japanese (12 lists per language). Each list contains all instances of a particular semantic class in a certain language, and each instance contains a set of synonyms (e.g., USA, America).

Since the output of our system is a ranked list of extracted instances, we choose mean average precision (MAP) as our evaluation metric. MAP is commonly used in the field of Information Retrieval for evaluating ranked lists because it is sensitive to the entire ranking and it contains both recall and precision-oriented aspects. The MAP for multiple ranked lists is simply the mean value of average precisions calculated separately for each ranked list. We define the average precision of a single ranked list as:

$$AvgPrec(L) = \frac{\sum_{r=1}^{|L|} \text{Prec}(r) \times \text{isFresh}(r)}{\text{Total \# of Correct Instances}}$$

where L is a ranked list of extracted instances, r is the rank ranging from 1 to $|L|$, Prec(r) is the precision at rank r, or the percentage of correct synonyms above rank r (inclusively). isFresh(r) is a binary function for ensuring that, if a list contains multiple synonyms of the same instance (or instance pair), we do not evaluate that instance (or instance pair) more than once. More specifically, the function returns 1 if a) the synonym at r is correct, and b) it is the highest-ranked synonym of its instance in the list; it returns 0 otherwise.

We evaluate the performance of each type of wrapper by conducting set expansion on the 36 datasets across three languages. For each dataset, we randomly select two seeds, expand them by bootstrapping ten iterations (where each iteration retrieves at most 200 web pages only), and evaluate the final result. We repeat this process three times for every dataset and report the average MAP for English, Japanese, and Chinese in Table 3. As illustrated, the more restrictive a wrapper is, the worse it performs. As a result, this indicates that further restrictions on wrappers of type 5 will not improve performance.

4 Set Expansion for Binary Relations

4.1 Identifying Wrappers for Binary Relations

We extend the wrapper construction algorithm described in Section 2.1 to support relational set expansion. The major difference is that we introduce a third type of context called the *middle* context that occurs between the left and right contexts of a wrapper for separating any two items. We execute the same algorithm as before, except that a seed instance in the algorithm is now a seed instance pair bracketing some middle context (i.e., "$s_1 \cdot middle \cdot s_2$").

Given some seed pairs (e.g., Ford and USA), the algorithm first locates the seeds in some given documents. For every pair of seeds located, it extracts their left, middle, and right contexts. The left and right contexts are inserted into their corresponding tries, while the middle context is inserted into a list. Every middle context is assigned a flag indicating whether the two instances bracketing it were found in the same or reversed order as the input seed pairs. Every entry in the seed instance list described previously now stores a pair of instances as one single string (e.g. "Ford/USA"). An id stored in a node now matches the index of a pair of instances as well as a middle context.

Shown below is a mock example document of which the seed pairs: Ford and USA, Nissan and Japan, Toyota and Japan are located (and boldfaced).

GtpKxH**nIsSaN**oKp**jaPaN**xjHJgleTuoLpBlcLBxKH
forDEFc**uSA**xkrpWNapnIkAAHO**FoRd**awHDa**USa**uoh
deQsKxHaCuRAoKpJapANxjHdIjWnOxo**TOyOTa**Vaq
jApaNzxKHAUdIEFcgErmANyxkrOyQKxH**ToYotA**oK
p**JApaN**xjHCRdmtqOVKxH**foRd**oKp**usA**xjHaJASzEi
nSfrla**FoRD**LMmp**uSa**ofwNLWxKH**tOYota**EFc**jAPan**
xkrHxQKzrHpoKdGEKtxKH**NisSan**EFc**JApAn**xkrEq

After performing the abovementioned procedures on this mock document, we now have context tries that are much more complicated than those illustrated in Figure 1, as well as a list of middle contexts similar to the one shown below:

id	Seed Pairs	r	Middle Context
0	Nissan/Japan	No	oKp
1	Nissan/Japan	No	EFc
2	Nissan/Japan	Yes	xkrHxQKzrHpoKd...
4	Toyota/Japan	No	oKp
6	Toyota/Japan	Yes	xjHdIjWnOxo
9	Ford/USA	No	EFc
13	Ford/USA	Yes	xkrpWNapnIkAAHO

where r indicates if the two instances bracketing the middle context were found in the reversed order as the input seed pairs. In order to find the maximally long contextual strings, the "Intersect" function in the set expansion pseudo-code presented in Table 2 needs to be replaced with the following:

Integers Intersect(Node n_1, Node n_2)
  Define $S = n_1.indexes \cap n_2.indexes$
  Return the largest subset s of S such that:
    Every index $\in s$ corresponds to same middle context

which returns those seed pairs that are bracketed by the strings associated with the two input nodes with the same middle context. A wrapper for relational set expansion, or *relational wrapper*, is defined by the left, middle, and right contextual strings. The relational wrappers constructed from the mock document given the example seed pairs are shown below. Notice that Audi/Germany and Acura/Japan are discovered.

Wrapper:	xKH[.1.]EFc[.2.]xkr
Content:	**audi/germany**, ford/usa, nissan/japan, toyota/japan
Wrapper:	KxH[.1.]oKp[.2.]xjH
Content:	**acura/japan**, ford/usa, nissan/japan, toyota/japan

Dataset ID	Item #1 vs. Item #2	Lang. #1	Lang. #2	Size	Complete?
US Governor	US State/Territory vs. Governor	English	English	56	Yes
Taiwan Mayor	Taiwanese City vs. Mayor	Chinese	Chinese	26	Yes
NBA Team	NBA Team vs. NBA Team	Chinese	English	30	Yes
Fed. Agency	US Federal Agency Acronym vs. Full Name	English	English	387	No
Car Maker	Car Manufacturer vs. Headquartered Country	English	English	122	No

Table 4: The five relational datasets for evaluating relational set expansion.

	Mean Avg. Precision					Precision@100				
Datasets	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
US Governor	97.4	89.3	89.2	89.3	89.2	55	50	51	50	50
Taiwan Mayor	99.8	95.6	94.3	91.3	90.8	25	25	24	23	23
NBA Team	100.0	99.9	99.9	99.9	99.2	30	30	30	30	30
Fed. Agency	43.7	14.5	5.2	11.1	5.2	96	55	20	40	20
Car Maker	61.7	0.0	0.0	0.0	0.0	74	0	0	0	0
Average	80.5	59.9	57.7	58.3	56.9	56	32	25	29	25

Table 5: Performance of various types of wrappers on the five relational datasets after first iteration.

	Mean Avg. Precision					Precision@100				
Datasets	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
US Governor	98.9	97.0	95.3	94.1	93.9	55	55	54	53	53
Taiwan Mayor	99.8	98.3	96.9	93.8	94.3	25	25	25	24	24
NBA Team	100.0	100.0	99.2	98.4	98.6	30	30	30	30	30
Fed. Agency	65.5	54.5	27.9	55.3	30.0	97	97	61	95	69
Car Maker	81.6	0.0	0.0	0.0	0.0	90	0	0	0	0
Average	89.2	70.0	63.9	68.3	63.4	59	41	34	40	35

Table 6: Performance of various types of wrappers on the five relational datasets after 10^{th} iteration.

4.2 Experiments with Binary Relations

For binary relations, we performed the same experiment as with unary relations described in Section 3. A relational wrapper is of type t if the wrapper's left and right context match t's constraint for left and right respectively, and also that the wrapper's middle context match both constraints.

For choosing the evaluation datasets for relational set expansion, we surveyed and obtained a dozen relationships, from which we randomly selected five of them and present in Table 4. Each dataset was then manually constructed. For the last two datasets, since there are too many items, we tried our best to make the lists as exhaustive as possible.

To evaluate relational wrappers, we performed relational set expansion on randomly selected seeds from the five relational datasets. For every dataset, we select two seeds randomly and bootstrap the relational set expansion ten times. The results after the first iteration are shown in Table 5 and after the tenth iteration in Table 6. When computing precision at 100 for each resulting list, we kept only the top-most-ranked synonym of every instance and remove all other synonyms from the list; this ensures that every instance is unique. Notice that for the "Car Maker" dataset, there exists no wrappers of types 2 to 5; thus resulting in zero performance for those wrapper types. In each table, the results indicate that character-based wrappers perform the best, while those HTML-based wrappers that require tight HTML bracketing of items (type 3 and 5) perform the worse.

In addition, the results illustrate that bootstrapping is effective for expanding relational pairs of items. As illustrated in Table 6, the result of finding translation pairs of NBA team names is perfect, and it is almost perfect for finding pairs of U.S. states/territories and governors, as well as Taiwanese cities and mayors. In finding pairs of acronyms and full names of federal agencies, the precision at top 100 is nearly perfect (97%). The results for finding pairs of car makers and countries is good as well, with a high precision of 90%. For the last two datasets, we believe that MAP could be improved by increasing the number of bootstrapping iterations. Table 7 shows some example wrappers constructed and instances extracted for wrappers of type 1.

Seeds: 台中縣 / 黃仲生, 嘉義縣 / 陳明文

URL: http://www.chinareviewnews.org/crn-webapp/attcoluOutline.jsp?coluid=40
Wrapper: 黨[.2.]宣布當選[.1.]長 (2005-12-03 1
URL: http://www.siongpo.com/20051204/localk.htm
Wrapper: 黨)；[.1.]：[.2.]（
URL: http://www.books.com.tw/exep/prod/booksfile.php?item=0010430903
Wrapper: 1 [.1.]長[.2.]最愛私房餐廳
URL: http://mag.udn.com/mag/vote2005/storypage.jsp?f_ART_ID=19060&pno=1
Wrapper: (國民黨)
 [.1.]長[.2.](
Content: 南投縣 / 蔡煌瑯, 台中市 / 胡志強, 台中縣 / 黃仲生, 台南縣 / 蘇煥智, 台東縣 / 劉櫂豪, 嘉義縣 / 陳明振
嘉義縣 / 陳明文, 宜蘭縣 / 陳定南, 彰化市 / 柯金德, 新竹市 / 林政則, 新竹市 / 鄭貴元, 新竹縣 / 鄭永金
桃園縣 / 朱立倫, 花蓮縣 / 盧博基, 花蓮縣 / 謝深山, 苗栗市 / 彭維彬, 苗栗縣 / 劉政鴻, 連江縣 / 陳雪生

Seeds: 鳳凰城太陽 / phoenix suns, 密爾瓦基公鹿 / milwaukee bucks

URL: http://www.xxlbasketball.com.tw/NBA-5/index.htm
Wrapper: >[.1.]
[.2.]</td>
URL: http://sandy0932266920.pixnet.net/blog/post/21155354
Wrapper: >[.1.]隊([.2.])
URL: http://www.nbamag.com.tw/html/oldmags/mag18.html
Wrapper: /updatemid.html">[.1.]
[.2.]<hr><td align=center>
URL: http://tw.myblog.yahoo.com/jw!sNocSW.QEkbxPfvMO5U8qk82LSGu/article?mid=23495
Wrapper:
[.1.]([.2.])
Content: 丹佛金塊 / denver nuggets, 錫拉丘茲民族隊 / syracuse nations, 亞特蘭大老鷹 / atlanta hawks, 休士頓火箭 / houston rockets
克裡夫蘭騎士 / cleveland cavaliers, 印第安那溜馬 / indiana pacers, 夏洛特山貓 / charlotte bobcats, 多倫多暴龍 / toronto raptors
奧蘭多魔術 / orlando magic, 孟菲斯灰熊 / memphis grizzlies, 密爾瓦基公鹿 / milwaukee bucks, 底特律活塞 / detroit pistons
明尼蘇達灰狼 / minnesota timberwolves, 沙加緬度國王 / sacramento kings, 波士頓塞爾蒂克 / boston celtics
波特蘭拓荒者 / portland trail blazers, 洛杉磯快艇 / l.a. clippers, 洛杉磯湖人 / l.a. lakers, 猶他爵士 / utah jazz

Seeds: **kentucky / steve beshear, north dakota / john hoeven**

URL: http://wikifoia.pbworks.com/Alaska-Governor-Sarah-Palin
Wrapper: Governor [.2.]">[.1.] Governor
URL: http://blogs.suntimes.com/sweet/2008/02/sweet_state_dinner_for_governo.html
Wrapper:

 The Honorable [.2.], Governor of [.1.]

URL: http://en.wikipedia.org/wiki/United_States_Senate_elections,_2010
Wrapper: " title="Governor of [.1.]">Governor <a href="/wiki/[.2.]" title="
URL: http://ballotbox.governing.com/2008/07/index.html
Wrapper: , [.1.]'s [.2.],
Content: alabama / bob riley, alaska / sarah palin, arizona / janet napolitano, arkansas / mike huckabee, california / arnold schwarzenegger, colorado / bill ritter, connecticut / mary jodi rell, delaware / ruth ann minner, florida / charlie crist, georgia / sonny perdue, hawaii / linda lingle, idaho / butch otter, illinois / rod blagojevich. . .

Seeds: **cia / central intelligence agency, usps / united states postal service**

URL: http://www1.american.edu/dccampus/links/whitehouse.html
Wrapper: [.2.]
URL: http://www.usembassy.at/en/us/gov.htm
Wrapper: /" target="_blank">[.2.] ([.1.]) -
URL: http://www.nationmaster.com/encyclopedia/List-of-United-States-federal-agencies
Wrapper: The [.2.] ([.1.]) is
URL: http://www.nationmaster.com/encyclopedia/List-of-United-States-federal-agencies
Wrapper: [.1.]- <a href="/encyclopedia/[.2.]" onmouseover="pv(event, 2
Content: achp / advisory council on historic preservation, arc / appalachian regional commission, cftc / commodity futures trading commission, cia / central intelligence agency, cms / centers for medicare and medicaid services, exim bank / export import bank of the united states, ntrc / national transportation research center. . .

Seeds: **mazda / japan, venturi / france**

URL: http://www.jrfilters.com/filtres/index.php?lng=en
Wrapper: &page=filtres&lng=en">[.1.]   ([.2.])</option><option value="index.php?
URL: http://www.jrfilters.com/suspensions/index.php?famille=1&lng=en
Wrapper: &lng=en">[.1.]   ([.2.])</option><option value="index.php?famille=1&rubrique1
URL: http://www.street-car.net/forums/forumdisplay.php?f=10
Wrapper: ">[.1.] Country of origin:[.2.].
URL: http://www.allcarcentral.com/
Wrapper: file.html">[.1.],[.2.]
Content: abarth / italy, acura / japan, alfa romeo / italy, aston martin / england, auburn / usa, audi / germany, austin healey / england, austin / england, auto union / germany, balwin / usa, bandini / italy, bentley / england, bmw / germany, brabham / england, bricklin / usa, bristol / england, brm / england, bucciali / france. . .

Table 7: Examples of (type 1) wrappers constructed and instances (contents) extracted.

5 Related Work

In recent years, many research has been done on extracting relations from free text (e.g., (Pantel and Pennacchiotti, 2006; Agichtein and Gravano, 2000; Snow et al., 2006)); however, almost all of them require some language-dependent parsers or taggers for English, which restrict the language of their extractions to English only (or languages that have these parsers). There has also been work done on extracting relations from HTML-structured tables (e.g., (Etzioni et al., 2005; Nadeau et al., 2006; Cafarella et al., 2008)); however, they all incorporated heuristics for exploiting HTML structures; thus, they cannot handle documents written in other mark-up languages.

Extracting relations at character-level from semi-structured documents has been proposed (e.g., (Kushmerick et al., 1997),(Brin, 1998)). In particular, Brin's approach (DIPRE) is the most similar to ours in terms of expanding relational items. One difference is that it requires maximally-long contextual strings to bracket *all* seed occurrences. This technique has been experimentally illustrated to perform worse than SEAL's approach on unary relations (Wang and Cohen, 2007). Brin presented five seed pairs of author names and book titles that he used in the experiment (unfortunately, he did not provide detailed results). We input the top two seed pairs listed in his paper into the relational SEAL, performed ten bootstrapping iterations (took about 3 minutes), and obtained 26,000 author name/book title pairs of which the precision at 100 is perfect (100%).

6 Conclusions

In this paper, we have described in detail an algorithm for constructing document-specific wrappers automatically for set expansion. In the experimental results, we have illustrated that character-based wrappers are better suited than HTML-based wrappers for the task of set expansion. We also presented a method that utilizes an additional middle context for constructing relational wrappers. We also showed that our relational set expansion approach is language-independent; it can be applied to non-English and even cross-lingual seeds and documents. Furthermore, we have illustrated that bootstrapping improves the performance of relational set expansion. In the future, we will explore automatic mining of binary concepts given only the relation (e.g., "mayor of").

7 Acknowledgments

This work was supported by the Google Research Awards program.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *In Proceedings of the 5th ACM International Conference on Digital Libraries*, pages 85–94.

Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *In WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT98*, pages 172–183.

Michael J. Cafarella, Alon Y. Halevy, Daisy Z. Wang, Eugene W. 0002, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *PVLDB*, 1(1):538–549.

A. Carlson, J. Betteridge, E.R. Hruschka Junior, and T.M. Mitchell. 2009. Coupling semi-supervised learning of categories and relations. In *NAACL HLT Workshop on Semi-supervised Learning for Natural Language Processing*, pages 1–9. Association for Computational Linguistics.

Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165(1):91–134.

N. Kushmerick, D. Weld, and B. Doorenbos. 1997. Wrapper induction for information extraction. In *Proc. Int. Joint Conf. Artificial Intelligence*.

David Nadeau, Peter D. Turney, and Stan Matwin. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In Luc Lamontagne and Mario Marchand, editors, *Canadian Conference on AI*, volume 4013 of *Lecture Notes in Computer Science*, pages 266–277. Springer.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 113–120, Morristown, NJ, USA. Association for Computational Linguistics.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 801–808, Morristown, NJ, USA. Association for Computational Linguistics.

Partha P. Talukdar, Thorsten Brants, Mark Liberman, and Fernando Pereira. 2006. A context pattern induction method for named entity extraction. In *Tenth Conference on Computational Natural Language Learning (CoNLL-X)*.

Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *ICDM*, pages 613–622. IEEE Computer Society.

Richard C. Wang and William W. Cohen. 2007. Language-independent set expansion of named entities using the web. In *ICDM*, pages 342–350. IEEE Computer Society.

Richard C. Wang and William W. Cohen. 2008. Iterative set expansion of named entities using the web. In *ICDM*, pages 1091–1096. IEEE Computer Society.

Classifying Relations for Biomedical Named Entity Disambiguation

Xinglong Wang[†‡] **Jun'ichi Tsujii**[*†‡] **Sophia Ananiadou**[†‡]
[†]School of Computer Science, University of Manchester, UK
[‡]National Centre for Text Mining, UK
[*]Department of Computer Science, University of Tokyo, Japan
{xinglong.wang,j.tsujii,sophia.ananiadou}@manchester.ac.uk

Abstract

Named entity disambiguation concerns linking a potentially ambiguous mention of named entity in text to an unambiguous identifier in a standard database. One approach to this task is supervised classification. However, the availability of training data is often limited, and the available data sets tend to be imbalanced and, in some cases, heterogeneous. We propose a new method that distinguishes a named entity by finding the informative keywords in its surrounding context, and then trains a model to predict whether each keyword indicates the semantic class of the entity. While maintaining a comparable performance to supervised classification, this method avoids using expensive manually annotated data for each new domain, and thus achieves better portability.

1 Introduction

While technology on named entity recognition (NER) matures, many researchers in the field of information extraction (IE) gradually shifted their focus to more complex tasks such as named entity disambiguation and relation extraction. Both tasks are particularly important for biomedical text mining, which concerns automatically extracting facts from the exponentially growing biomedical literature (Hunter and Cohen, 2006). One type of facts is relations between biomedical named entities, such as disease-drug relation, gene-disease relation, protein-protein interaction (PPI), etc. To automatically extract these facts, advanced natural language processing techniques such as parsing have been adopted to analyse the syntactic and semantic structure of text. The idea is that linguistic structures between the interacting biological entities may have common characteristics that can be exploited by similarity measures or machine learning algorithms. For example, Erkan et al. (2007) used the shortest path between two genes according to edit distance in a dependency tree to define a kernel function for extracting gene interactions. Miwa et al. (2008) comparably evaluated a number of kernels for incorporating syntactic features, including the bag-of-word kernel, the subset tree kernel (Moschitti, 2006) and the graph kernel (Airola et al., 2008), and they concluded that combining all kernels achieved better results than using any individual one. Miyao et al. (2008) used syntactic paths as one of the features to train a support vector machines (SVM) model for PPIs and also discussed how different parsers and output representations affected the end results.

Another crucial IE task is named entity disambiguation, which concerns grounding mentions of named entities in text to unambiguous *concepts* as defined in some standard dictionary or database. For instance, given a search term *Python*, users may like to see the results grouped into the following categories: *a type of snake*, *a programming language*, or *a film* (Bunescu and Paşca, 2006). One approach to such lexical disambiguation tasks is supervised classification. However, such techniques suffer from the knowledge acquisition bottleneck, meaning that manually annotating training data is costly and can never satisfy the need by the machine learning algorithms. In addition, supervised techniques may not yield reliable results when the distributions of the semantic classes are different in the training and test datasets (Agirre and Martinez, 2004; Koeling et al., 2005). For example, on the task of word sense disambiguation, a model trained on a dataset where the predominant sense of the word *star* is "heavenly body", may not work well on text mainly composed of entertainment news. Such problems are also major concerns when developing a system to disambiguate biomedical named entities (e.g., protein,

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1513–1522,
Singapore, 6-7 August 2009.

gene, and disease), for which some researchers rely on hand-crafted rules in addition to a small amount of training data (Morgan and Hirschman, 2007; Hakenberg et al., 2008).

This paper proposes a new disambiguation method that, instead of classifying each individual occurrence of an entity, it classifies pair-wise relations between the entity mention in question and the "*cue words*" in its adjacent context, where each cue word is assumed to bear a semantic class. We then select the cue word that has a positive relation with the entity, and pass its semantic tag to it. While an individual entity mention may belong to a large number of semantic classes, a relation can only take one of two values: positive or negative, hence transforming a complex multi-classification problem into a less complicated binary classification task. The remainder of the paper is organised as follows: Section 2 proposes the disambiguation method and Section 3 introduces the task of disambiguating the model organisms of biomedical named entities. Section 4 describes in detail our proposed method and also a number of baseline systems for comparison purposes. Section 5 shows the evaluation results and discusses the advantages and drawback of our system, and we finally conclude in Section 6.

2 Disambiguation as Relation Classification

The named entity disambiguation task is defined as follows: given a mention of a named entity in text, we automatically assign a semantic tag d to it, where $d \in D$, and D is a pre-compiled dictionary with $|D|$ entries. When $|D|$ is small, the problem can be approached by supervised classification. For example, to determine whether an occurrence of an entity is a protein, a gene or an RNA, Hatzivassiloglou et al. (2001) compared performance of 3 supervised classification methods and reported results near the human agreement rate. Nevertheless, when $|D|$ is large (e.g., > 100), the performance of classification may decrease, especially when the distribution of d in training dataset differs from that in the test set. In other words, when $|D|$ is large, named entity disambiguation becomes a multi-class classification task on heterogeneous and imbalanced datasets, which is challenging for a machine learning model to learn to discriminate enough between the semantic classes (Japkowicz, 2000).

We propose an alternative method for named entity disambiguation. Intuitively, in the surrounding context of an ambiguous entity, one can often find "cue words" that are informative indicators of the entity's semantic category. These cue words are provided by authors to remind readers the semantic identity of a named entity. For example, in an article about protein p53, phrase "human protein p53" may be mentioned, where both *human* and *protein* contain semantic information regarding p53: *human* indicates the model organism of p53, and *protein* suggests the type of this entity. Such cue words may occur infrequently in the training data, making it difficult for machine learning classifiers to capture.

Our method exploits this observation. Given a sentence, let E be the set of 'target' entities (e.g., *p53*) and W of the 'cue' words (e.g., *human*) that co-occur in a sentence, we define a relation as a pair $r = \langle e, w \rangle$, where $e \in E$ and $w \in W$, and r is a positive relation if e belongs to the semantic class indicated by w, and is a negative one if not. Then we can disambiguate e by accomplishing the following steps: 1) identify W and build a set of relations $R = \{\langle e, w_i \rangle | w_i \in W, i = 1, 2, .., n\}$, where n is the size of W; and 2) classify every $r \in R$ and assign the semantic tag of w_j to e such that $r_j = \langle e, w_j \rangle$ is positive. The first task can be tackled by a dictionary lookup, or by an NER system, if manually annotated data is available. The second is essentially a binary relation classification task, and in this work, we use an SVM model exploiting bag-of-word and syntactic features.

3 Species Disambiguation

We show the performance of the proposed method on a task of resolving one major source of ambiguity in protein and gene entities: model organisms. Model organisms are species studied to understand particular biological phenomena. Biological experiments are often conducted on one species, with the expectation that the discoveries will provide insight into the workings of others, including humans, which are more difficult to study directly. From viruses, prokaryotes, to plants and animals, there are dozens of organisms commonly used in biological studies, such as *E. coli*, *Drosophila*, *Homo sapiens*, and hundreds more are frequently mentioned in biological research papers. In biomedical articles, entities of different species are commonly referred to us-

ing the same name, causing great ambiguity. For example, searching a protein sequence database, RefSeq[1] with query "*tumor protein p53*" resulted in over 100 proteins, as the name is shared by many organisms.

The importance of distinguishing model organisms has been recognised by the community of biomedical text mining. Chen et al. (2005) collected gene names from various source databases and calculated intra- and inter-species ambiguities. Overall, only 25 (0.02%) official symbols were ambiguous within the organisms. However, when official symbols from 21 organisms were combined, the ambiguity increased substantially to $21,279$ (14.2%) symbols. Hakenberg et al. (2008) showed that species disambiguation is one of the most important steps for term normalisation and identification, which concerns automatically associating mentions of biomedical entities in text to unique database identifiers (Morgan et al., 2008). Also, the task of extracting PPIs in the recent BioCreative Challenge II workshop (Hirschman et al., 2007) requires protein pairs to be recognised and normalised, which inevitably involves species disambiguation.

More specifically, given a text, in which mentions of biomedical named entities are annotated, a species disambiguation system automatically assigns a *species identifier*, as in a standard database of model organisms, to every entity mention. The types of biomedical named entities concerned in this study are protein, gene, protein complex and mRNA/cDNA, and we used identifiers from the NCBI Taxonomy of model organisms.[2] The work focuses on species disambiguation and assumes that the entities are already identified. In practice, an automated named entity recogniser (e.g., ABNER (Settles, 2005)) should be used before applying the systems.

4 Approaches

This section describes a number of approaches to species disambiguation, highlighting the relation classification method proposed in Section 2.

4.1 Heuristics Baselines

The cue words for species are words denoting names of model organisms (e.g., *mouse* as in phrase "mouse p53"). Another clue is the presence of the species-indicating prefixes in gene and protein names. For instance, prefix 'h' in entity "hSos-1" suggests that it is a *human* protein. Throughout this paper, we refer to such cue words (e.g., *mouse*, *hSos-1*) as "species words". Note that a species "word" may contain multiple tokens (e.g., *E. Coli*).

We encoded this knowledge in a rule-based species tagging system (Wang and Grover, 2008). The system takes a 2-step approach. First, it marks up species words in the document using a species-word detection program,[3] which searches every word in a dictionary of model organisms and assigns a species ID to the word if a match is found. The dictionary was built using the NCBI taxonomy[4] and the UniProt controlled vocabulary of species,[5] and in total it contains 420,224 species words for 324,157 species IDs. When species words are identified, we disambiguate an entity mention using *one of* the following rules:

1. *previous species word*: If the word preceding an entity is a species word, assign the species ID indicated by that word to the entity.

2. *species word in the same sentence*: If a species word and an entity appear in the same sentence, assign its species ID to the entity. When more than one species word co-occurs in the sentence, priority is given to the species word to the entity's left with the smallest distance. If all species words occur to the right of the entity, take the nearest one.

3. *majority vote*: assign the most frequently occurring species ID in the document to all entity mentions.

It is expected that the first rule would produce good precision. However, it can only disambiguate the fraction of entities that happen to have a species word to their *immediate* left. The second rule relaxes the first by allowing an entity to take the species indicated by its nearest species word in the same sentence, which should increase recall but decrease precision. Statistics from our dataset (see Section 5.1) show that only 5.68% entities can potentially be resolved by rule 1 and 22.16% by rule 2, while the *majority* rule can tackle every entity mention in the dataset.

[1] http://www.ncbi.nlm.nih.gov/RefSeq

[2] http://www.ncbi.nlm.nih.gov/sites/entrez?db=taxonomy

[3] The species word detector identifies the cue words and was used in all the systems studied in this paper. We could not properly evaluate the detector due to the lack of manually annotated data. Its performance, however, would not affect the comparative evaluation results, and improvement to species word detection should increase the performance of these disambiguation systems.

[4] ftp://ftp.ncbi.nih.gov/pub/taxonomy/

[5] http://www.expasy.ch/cgi-bin/speclist

4.2 Supervised Classification

The disambiguation problem can be approached as a classification task. Given an entity mention and its surrounding context, a machine learning model classifies the entity into one of the classes, where each class corresponds to a species ID. We carried out experiments with two classification methods: multi-class classification and one-class classification, where a maximum entropy model[6] was used for the former and SVM-light[7] for the latter. In one-class classification, we trained a series of binary SVM classifiers, each constructing a separating hyperplane that maximises the margin between the instances of one specific species (i.e., the target class) and a set of randomly selected instances of other species (i.e., the outlier class). We used equal numbers of instances for both classes in training. The following types of features were used in both multi-class and one-class experiments, where the values of n were set empirically by cross-validation on the training data:

- *leftContext* The n word lemmas to the left of the entity ($n = 200$).
- *rightContext* The n word lemmas to the right of the entity ($n = 200$).
- *leftSpeciesIDs* The n species IDs to the left of the entity (with order, $n = 5$).
- *rightSpeciesIDs* The n species IDs to the right of the entity (with order, $n = 5$).
- *leftNouns* The n nouns to the left of the entity (with order, $n = 2$).
- *leftAdjs* The n adjectives to the left of the entity (with order, $n = 2$).
- *leftSpeciesWords* The n species word forms to the left of the entity ($n = 5$).
- *rightSpeciesWords* The n species word forms to the right of the entity ($n = 5$).
- *firstLetter* The first character of the entity itself (e.g., 'h' in *hP53*).
- *documentSpeciesIDs* All species IDs that occur in the document in question.
- *useStopWords* filter out function words.
- *useStopPattern* filter out words consisting only of digits and punctuation characters.

Feature selection was also carried out for the one-class classification experiments. We compared two feature selection methods that reportedly work well on the task of text classification: information gain (IG) (Yang and Pedersen, 1997) and Bi-Normal separation (BNS) (Forman, 2003). IG measures the decrease in entropy when the feature is given vs. absent, and is defined as: $IG(Y|X) = H(Y) - H(Y|X)$ where $H(Y)$ is the uncertainty about the value of Y (i.e., Y's entropy), and $H(Y|X)$ is Y's conditional entropy given X. The BNS is defined as: $|F^{-1}(x) - F^{-1}(y)|$, where F^{-1} is the standard Normal distribution's inverse cumulative probability function, namely, *z-score*; x is the ratio between the number of positive cases containing the feature in question, and the total number of positive cases; and y is the ratio between the number of negative cases containing the feature, and the total number of negative cases.

We computed a weight for each feature and then ranked the features according to their weight, with respect to each feature selection method. The top 10% features were used in training. Given a test instance, the one-class classification method first counts the species words in the document that the instance appears in, and then applies in sequence the binary models of each occurring species, starting from the most frequent one. For example, if a document contains 5 occurrences of *human* and 3 *mouse*, we first apply the *human* species model to judge whether an entity mention is of *human* species, and only if not, the *mouse* model was applied. The most-frequent species in the document was used as backup when none of the binary models gives positive answers.

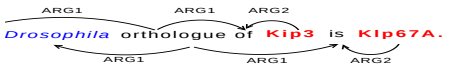


Figure 1: Predicate argument structure (PAS).

4.3 Relation Classification

4.3.1 Overview

As for the proposed relation classification method, in the training phase, we first selected the sentences in which an entity mention and a species word co-occur, and constructed pair-wise entity-species relations. We then assigned each relation a binary label: a relation is positive if the species ID inferred from the species word matches the gold-standard species annotation on the entity, and is negative otherwise. For example, for the sentence shown in Figure 1, where *Drosophila* is a species word, and Kip3 and Klp67A are proteins, relation ⟨Kip3, *Drosophila*⟩ is a negative instance and the

[6] http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

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

pair ⟨Klp67A, *Drosophila*⟩ is a positive one.[8]

For each relation, a vector of features were extracted. We followed the PPI extraction method described in (Miyao et al., 2008), where two types of features were used for a SVM classifier. The first was bag-of-word features, i.e., the words before, between and after the pair of entities, where the words were lemmatised. We added an additional feature of the distance between the entity and the cue word. The other type was syntactic features obtained from parsers. For bag-of-word features, a linear kernel was used, and for syntactic ones, a subset tree kernel (Moschitti, 2006) was adopted. The syntactic features were represented in a flat tree format. Figure 2 shows such a feature for the negative instance ⟨Kip3, *Drosophila*⟩ from Figure 1. Note that all species words (e.g., *Drosophila*) were normalised to "SPECIESWORD", and entities (e.g., Kip3) to "ENTITY", which not only reduces the noise in the feature set, but also makes the model more species-generic. From the training dataset (see Section 5.1), $25,413$ relations were extracted, of which 63.3% were positive.

```
(ENJU(noun_arg1(SPECIESWORD orthologue))
(prep_arg12(of orthologue))
(prep_arg12(of ENTITY)))
```

Figure 2: A syntactic feature obtained from the ENJU parser.

To identify the species of an entity in unseen text, we first parsed the sentence, and then listed all pairs of species words and entities as relations. Having extracted the bag-of-word and syntactic features from the instance, the trained model was applied to judge whether each species-entity relation was positive. The entity mention in a positive relation would be tagged with the ID indicated by the species word, while the mentions in negative relations would be left untagged. The next section describes in detail how we extracted the syntactic features from text.

4.3.2 Syntactic Features

Given a sentence, a natural language parser automatically recognises its syntactic structure and outputs a parse tree, in which nodes represent words or syntactic constituents. A path between a pair of nodes can be interpreted as a syntactic relation between sentence units, which was proved useful to infer biological relations (e.g., Airola et al., 2008; Miwa et al., 2008).

Parser	Input	Output
C&C	POS-tagged	GR
ENJU	POS-tagged	PAS
ENJU-Genia	POS-tagged	PAS
Minipar	Sentence-detected	Minipar
RASP	Tokenised	GR
Stanford	POS-tagged	SD
Stanford-Genia	POS-tagged	SD

Table 1: Parsers and their input and output format

We experimented with the following parsers (summarised in Table 1):

- **Dependency parsers** identify one word as the head of a sentence and all other words are either a dependent of that word, or else dependent on some other word that connects to the headword through a sequence of dependencies. We used Minipar (Lin, 1998) and RASP (Briscoe et al., 2006) for the experiments;
- **Constituent-structured parsers** split a sentence into syntactic constituents such as *noun phrases* or *verb phrases*. We used the Stanford parser (Klein and Manning, 2003), and also a variant of the Stanford parser (i.e., Stanford-Genia), which was trained on the GENIA treebank (Tateisi et al., 2005) for biomedical text;
- **Deep parsers** aim to compute in-depth syntactic and semantic structures based on syntactic theories such as HPSG (Pollard and Sag, 1994) and CCG (Steedman, 2000). We used the C&C parser (Clark and Curran, 2007), ENJU (Miyao and Tsujii, 2008), and a variant of ENJU (Hara et al., 2007) adapted for the biomedical domain (i.e., ENJU-Genia);

There were a number of practical issues to consider when using parsers for this task. Firstly, before parsing, the text needs to be linguistically pre-processed, and the quality of this process has a significant impact on parsers' performance. The pre-processing steps include sentence boundary detection, tokenisation and part-of-speech (POS) tagging, all of which can be tricky especially when applied to biomedical text (Grover et al., 2003). To avoid the noise that can be introduced in the pre-processing steps and to concentrate on evaluating the performance of the parsers, we used the same pre-processing tools (Alex et al., 2008a)[9] whenever possible. The middle column in Table 1 shows how the input text was linguistically pre-processed with respect to each parser. A POS-tagged text implies that it was also sentence boundary detected and tokenised Except for

[8] *Orthologues* are genes/proteins in different species but have similar sequences. In this example it implies that Klp67A is a *Drosophila* protein but Kip3 is not.

[9] These particular tools were chosen because they were adopted to pre-process the ITI-TXM dataset, which we used in our study.

RASP and Minipar, all parsers took POS-tagged text as input. RASP requires POS tags and punctuation labels that were derived from the CLAWS-7 tagset,[10] whereas our dataset uses POS labels from the Penn Treebank tagset (Marcus et al., 1994). As RASP does not recognise the Penn tagset, we used its build-in POS tagger. Minipar, on the other hand, does not support input of tokenised or POS-tagged text, and therefore took split sentences as input.

Secondly, the output representations of the parsers are different and we preferred a format that depicts relations between words instead of syntactic constituents. In total, 4 representations were used: grammatical relation (GR) (Briscoe et al., 2006), Stanford typed dependency (SD) (de Marneffe et al., 2006), Minipar's own representation (Lin, 1998), and ENJU's predicate-argument structure (PAS). All the above representations define relations of words in triples, where a dependency triple (i.e., GR, SD and Minipar) consists of head, dependent and relation, and a PAS triple contains predicate, argument, and relation. Figure 1 shows a sentence parsed by ENJU in PAS representation. The right-most column in Table 1 lists the output representation of each parser. A syntactic path between an entity and a species word was represented by a sequence of triples, each following the order of head-dependent or predicate-argument. These paths were used as syntactic features for the SVM classifier.

4.4 Spreading Strategies

Except for the *majority vote* rule, the approaches described in Sections 4.1 and 4.3 were expected to yield low recall, because they can only detect intra-sentential relations, and therefore only be applied to the entities having at least one species word appearing in the same sentence.

Since our aim is to disambiguate as many entity mentions as possible, we would like to "spread" the decisions from the disambiguated mentions to their "relatives" in the same document. We define an entity mention $\bar{e}$ as another mention e's relative under either of the following conditions: a) if $\bar{e}$ has the same surface form with e; or, b) if $\bar{e}$ is an abbreviation or an antecedent of e, where abbreviation/antecedent pairs were detected using the algorithm described in (Schwartz and Hearst, 2003). Given the set of disambiguated mentions, we then "spread" their species IDs to their relatives in the same document. After this process, the mentions that do not have any disambiguated relatives would still be missed by the system. In such cases, we used a "default" species, as determined by the rule of *majority vote* (see Section 4.1).

5 Evaluation

5.1 Data and Ontology

The species disambiguation experiments were conducted using the ITI-TXM corpus (Alex et al., 2008b), a collection of *full-length* biomedical research articles manually annotated with linguistic and biomedical information for developing automatic information extraction systems. The corpus contains two datasets covering slightly different domains: enriched protein-protein interaction (EPPI) and tissue expression (TE). Whenever possible, protein, protein complex, gene, and mRNA/cDNA entities were tagged with NCBI Taxonomy IDs, denoting their species, and it was the species annotation that this study used.

The EPPI and TE datasets have different distributions of species. The entities in EPPI belong to 118 species with *human* being the most frequent at 51.98%. In TE, the entities are across 67 species and *mouse* is the most frequent at 44.67%.[11] The inter-annotator agreement of species annotation on EPPI and TE are 86.45% and 95.11%, respectively.

The species disambiguation systems were developed on the training portions of the EPPI and TE corpora, each containing 221 articles, and evaluated on a dataset combining the development test (DEVTEST) datasets of EPPI and TE, containing 58 and 48 articles, respectively. The combined training dataset contains $96,992$ entity mentions belonging to 138 model organisms, while the DEVTEST dataset contains $23,118$ entities of 54 species. The diversity of model organisms in this corpus highlights the fact that a primary consideration when developing a species disambiguation system is its ability to distinguish a wide range of species with minimal additional manual effort.

5.2 Results

5.2.1 Evaluation Metrics

The evaluation was carried out on the DEVTEST dataset, and the systems are compared using av-

[10] http://ucrel.lancs.ac.uk/claws7tags.html

[11] These figures were obtained from the training split of the datasets.

	micro-avg.	macro-avg.
Maxent	70.48 / 70.48 / 70.48	10.07 / 10.00 / 9.85
SVM	62.24 / 59.35 / 60.76	14.70 / 17.11 / 15.01
SVM (IG)	65.20 / 61.06 / 63.06	14.90 / 19.53 / 16.09
SVM (BNS)	43.61 / 42.63 / 43.11	11.99 / 10.05 / 9.34

Table 2: Evaluation results of the classification systems on DEVTEST (precision/recall/F1-score, in %)

eraged precision, recall and F1 scores over all species. In more detail, for each model organism that appears in the DEVTEST dataset, we collect two lists of entity mentions of that species: one from the gold-standard DEVTEST dataset, and the other from the output of a disambiguation system. Then the list of system output is compared against the gold-standard list to obtain precision, recall and F1 score. For each system, the scores obtained from all species are averaged using micro-average and macro-average. The micro-average is the mean of the summation of contingency metrics for all model organisms, so that scores of the more frequent species influence the mean more than those of less frequent ones. The macro-average is the mean of precision, recall, or F1 over all labels, thus attributing equal weights to each species, and measuring a system's adaptability across different model organisms.

5.2.2 Evaluation Results

First of all, Table 2 shows the results of the classification methods described in Section 4.2. The multi-classification system using a maximum entropy model (Maxent) yielded the highest overall micro-averaged F1. Among the SVM-based systems, the one using IG feature selection achieved better performance. In particular, it outperformed the Maxent model in term of macro-averages. The performance of the SVM model with BNS feature selection is disappointing, perhaps because the occurrences of a feature in each instance are not normally distributed. As the Maxent system obtained better results, it was used to compare with other disambiguation systems.

Table 3 shows the results of a number of methods described in the previous sections. The methods are categorised into 4 groups: rule-based baseline systems, a Maxent classification model, relation-classification methods, and a hybrid system. The difference between the relation classification systems is the features adopted. Rel-Context was trained on only bag-of-word and distance features, whereas each other system also used syntactic features provided by a specific parser. For example, the Rel-RASP system identifies an entity's species by finding positive relations between the entity and its neighbouring species words, using features including bag-of-word, distance, and dependency paths generated by RASP. The hybrid system (Hbrd) ran the Rel-ENJU-Genia system on top of the outcome of Maxent. When a conflict occurs, the species ID is chosen by Rel-ENJU-Genia. The idea is that the relation classification system is more accurate than Maxent when it is applicable, and hence would improve precision on disambiguating the species with few or no training instances.

Without spreading (shown in the "NO SPRD" columns of Table 3), most of the rule-based and relation classification systems only work on a subset of DEVTEST, resulting in low recall: Rule-Sp works on the small proportion of entities (5.68%) with a preceding species word, while the other systems only work on the collection of sentences containing at least one species word and one entity, which covers 4.60% sentences and 22.16% entity mentions. Rule-Majority, Maxent, and Hbrd, on the other hand, apply to all entity mentions, and therefore they are only compared against the others when spreading was applied.

The results shown in the "NO SPRD" columns can be viewed as a comparative evaluation of the usefulness of the syntactic features supplied by the parsers on this particular task. The rule-based systems set high baselines: Rule-Sp produced good precision and Rule-SpSent achieved the highest micro-averaged F1, thanks to its high coverage, which is also an upperbound of recall for the relation classification systems. Nevertheless, it is encouraging that the relation classification systems obtained higher precision than Rule-SpSent, which is important, considering the decisions will be transfered to the untagged entity mentions across the document. Indeed, as shown in the SPRD columns in Table 3, most relation classification systems outperformed the Rule-SpSent baseline when spreading was used. The scores of the systems using different parser outputs only vary slightly. Rel-Context, on the other hand, surpassed others in terms of micro-averaged precision, while sacrificing micro-averaged recall and macro-averaged scores.

Next, the SPRD columns in Table 3 show the results when the spreading rules were applied, which

METHOD	NO SPRD (micro-avg)	NO SPRD (macro-avg)	SPRD (micro-avg)	SPRD (macro-avg)
Rule-Majority	N/A	N/A	66.14 / 61.99 / 64.00	16.76 / 21.75 / 18.08
Rule-Sp	88.96 / 5.02 / 9.51	33.77 / 8.55 / 10.18	66.96 / 63.41 / 65.13	28.25 / 30.65 / 27.00
Rule-SpSent	80.82 / 16.88 / 27.93	43.16 / 28.85 / 24.73	67.34 / 63.22 / 65.21	22.65 / 26.42 / 23.10
Maxent	N/A	N/A	70.48 / 70.48 / 70.48	10.07 / 10.00 / 9.85
Rel-Context	90.04 / 3.71 / 6.13	15.23 / 4.45 / 4.90	67.34 / 63.22 / 65.21	22.65 / 26.42 / 23.10
Rel-C&C	82.79 / 16.14 / 27.02	43.97 / 29.56 / 25.60	66.59 / 63.64 / 65.08	32.29 / 33.20 / 29.14
Rel-ENJU	83.39 / 15.87 / 26.66	46.89 / 29.88 / 25.95	68.28 / 65.02 / 66.61	31.82 / 34.08 / 29.67
Rel-ENJU-Genia	83.54 / 15.74 / 26.49	44.13 / 29.93 / 25.78	68.91 / 65.45 / 67.13	32.00 / 34.87 / 30.21
Rel-Minipar	81.82 / 16.27 / 27.14	43.63 / 27.88 / 24.15	67.98 / 63.77 / 65.81	31.83 / 33.93 / 29.44
Rel-RASP	81.67 / 16.10 / 26.90	43.95 / 28.92 / 25.03	66.62 / 64.08 / 65.33	32.66 / 33.54 / 29.80
Rel-Stanford	82.75 / 16.10 / 26.95	44.05 / 29.49 / 25.92	66.81 / 63.81 / 65.28	32.67 / 33.03 / 29.45
Rel-Stanford-Genia	82.22 / 16.04 / 26.84	43.37 / 29.40 / 25.22	66.85 / 63.64 / 65.21	32.72 / 32.29 / 28.64
Hbrd	N/A	N/A	74.15 / 73.26 / 73.70	43.98 / 37.47 / 31.80

Table 3: Evaluation results of the species disambiguation systems on DEVTEST (precision/recall/F1-score, in %)

effectively improved recall (see Section 5.2.3 for discussion on statistical significance tests on the results). The Maxent system achieved very good micro-averaged precision, but low macro-averaged scores. In fact, as shown in Table 4, Maxent can only disambiguate 7 species (out of a total of 54) that have relatively large amount of training instances,[12] and failed completely on other species. This suggests that Maxent may not be able to generate good micro-averaged scores when applied to a dataset where the dominant species are different from those in the training set. On the other hand, the relation-classification approaches have a clear advantage over Maxent as measured by macro-averaged scores. As shown in Table 4, Rel-ENJU-Genia worked well on most of the species, displaying its good adaptability, while achieving comparable micro-averaged F1 to Maxent. Overall, Hbrd, which combines the strengths of relation classification and the Maxent classification model, obtained the highest points as measured by every metric.

Species Name	Pct	Mxt	Rel	Hbrd
H. sapiens	50.13%	76.25	65.33	79.51
M. musculus	13.99%	66.41	58.29	68.27
X. tropicalis	7.35%	64.80	77.72	71.39
D. melanogaster	6.34%	93.17	78.46	95.15
S. cerevisiae	4.79%	90.12	83.32	87.68
R. norvegicus	2.97%	44.04	38.69	51.77
T. aestivum	2.62%	0.00	89.68	23.35
P. americana	2.27%	0.00	98.50	7.76
C. elegans	2.08%	96.83	95.88	97.50
H. herpesvirus 5	1.58%	0.00	54.46	4.27
R. virus	1.45%	0.00	28.54	6.45
H. spumaretrovirus	1.17%	0.00	99.37	2.49
...	...	...	...	...
Macro-average		9.85	30.21	31.80
Micro-average		70.48	67.13	73.70

Table 4: The micro-averaged F1 scores (%) of Maxent (Mxt), Rel-ENJU-Genia with spreading (Rel), and Hbrd with respect to each of the most frequent 12 species in DEVTEST.

5.2.3 Statistical Significance

To see whether our methods significantly improved the baseline systems, we performed randomisation tests (Noreen, 1989; Yeh, 2000) on some of the results shown in Table 3. The intuition of randomisation test is as follows: when comparing two systems (e.g., A and B), we erase the labels "output of A" or "output of B" from all observations. The null hypothesis is that there is no difference between A and B, and thus any response produced by one of the systems could have as likely come from the other. We shuffle these responses R times, reassign each response to A or B and see how likely such a shuffle produces a difference in the metric of interest that is at least as large as the difference observed when using A and B on the test data. Let r denote the number of times that such a difference occurred, then as $R \rightarrow \infty$, $\frac{r+1}{R+1}$ approaches the significance level. In our case, the metrics tested were micro- and macro-averaged precision, recall and F1.

Following this procedure, we tested whether the improvements made by a relation classification based system (i.e., Rel-ENJU-Genia with SPRD) and the hybrid system (i.e., Hbrd) over the baseline systems were statistically significant. We carried out approximate randomisation with 10,000 shuffles and the test results are shown in Table 5. The numerical figures in the cells are differences in precision, recall and F1 between a pair of systems. The significance levels (i.e., p-values) are indicated by superscript marks, whose corresponding values are displayed in Table 6. For exam-

[12] The following 7 species occur most frequently in the training set: H. sapiens (43.25%), M. musculus (27.05%), R. norvegicus (5.35%), S. cerevisiae (3.98%), X. tropicalis (3.56%), D. melanogaster (3.33%) and C. elegans (0.94%).

		Rule-Majority	Rule-Sp	Rule-SpSent	Maxent
Rel	micro-avg	2.77*/3.46*/3.13*	1.95*/2.04*/2.00*	1.57*/2.22*/1.92*	-1.57* / -5.02* / -3.35*
	macro-avg	15.24*/13.12*/12.13*	3.75[a]/4.21[a]/3.20[a]	9.35*/8.44*/7.10*	21.92*/24.87*/20.35*
Hbrd	micro-avg	8.01*/11.27*/9.70*	7.19*/9.85*/8.57*	6.81*/10.04* /8.49*	3.67*/2.78*/2.82[b]
	macro-avg	27.22*/15.72[c]/13.72[d]	15.73*/6.82[e]/4.80[f]	21.33*/11.05[g] / 8.70[h]	33.91[i]/27.47*/21.95*

Table 5: Results of paired randomisation tests on whether Rel-ENJU-Genia with SPRD (Rel) and Hbrd significantly improved the baseline systems. The numerical figures in the cells show the differences between the two systems as measured by precision/recall/F1 in percentage. The superscript marks indicate the significance levels and are explained in Table 6.

ple, the difference in micro-averaged precision between Rel-ENJU-Genia and Rule-Majority on the test data was 2.77%, and in 10,000 approximate randomisation trials, there was zero times[13] that Rel-ENJU-Genia's micro-averaged precision is greater than Rule-Majority's by at least 2.77% ($p < 0.0001$).

MARK	VALUE	MARK	VALUE
*	$p < 0.0001$	a	$p < 0.06$
b	$p < 0.002$	c	$p < 0.0003$
d	$p < 0.0002$	e	$p < 0.03$
f	$p < 0.05$	g	$p < 0.003$
h	$p < 0.005$	i	$p < 0.07$

Table 6: p-values.

The test results confirmed that, the improvements made by Hbrd are statistically significant with at least 95% confidence as measured by all metrics except for macro-averaged precision. The relation classification approach achieved significantly lower performance than Maxent in terms of micro-averaged scores (hence the "-" sign in the corresponding cell in Table 5), but in all other cases it can reject the null hypothesis with very high confidence (i.e., $p < 0.0001$).

6 Conclusions and Future Work

This paper proposes a method that tackles a complex disambiguation problem by breaking it into two cascaded simpler tasks of cue word discovery and binary relation classification. We evaluated the method on the task of disambiguating the model organisms of biomedical named entities, along with a number of other approaches. As measured by micro-averaged F1 score, a supervised classification approach (Maxent) yielded the second best result. However, it can only disambiguate a small number of species that have abundant training instances. With spreading rules, a relation classification system (Rel-ENJU-Genia) trained on word and syntactic features from ENJU-Genia also obtained good micro-averaged F1, while surpassing Maxent significantly in terms of macro-averaged scores. Combining these two systems achieved the best overall performance. Nevertheless, we combined the two methods in a rather crude way, leaving ample room for exploring better strategies in the future.

One drawback of the relation classification systems is that they can not cover all entity mentions but only the ones with informative keywords co-occurring in the same sentence. We overcame the drawback by using spreading rules. For some applications, however, it may be sufficient to make predictions exclusively for cases where the systems are applicable. Also, the predictions with high confidence can be used as seed training material for automatically harvesting more training data.

[13]The numbers of times are not shown in Table5 for brevity.

Acknowledgments

The work reported in this paper is funded by Pfizer Ltd.. The UK National Centre for Text Mining is funded by JISC. The ITI-TXM corpus used in the experiments was developed at School of Informatics, University of Edinburgh, in the TXM project, which was funded by ITI Life Sciences, Scotland.

References

E. Agirre and D. Martinez. 2004. Unsupervised WSD based on automatically retrieved examples: The importance of bias. In *Proceedings of EMNLP*.

A. Airola, S. Pyysalo, J. Björne, T. Pahikkala, F. Ginter, and T. Salakoski. 2008. A graph kernel for protein-protein interaction extraction. In *Proceedings of BioNLP*.

B. Alex, C. Grover, B. Haddow, M. Kabadjov, E. Klein, M. Matthews, S. Roebuck, R. Tobin, and X. Wang. 2008a. Assisted curation: does text mining really help? In *Proceedings of the Pacific Symposium on Biocomputing*.

B. Alex, C. Grover, B. Haddow, M. Kabadjov, E. Klein, M. Matthews, S. Roebuck, R. Tobin, and X. Wang. 2008b. The ITI TXM corpus: Tissue expression and protein-protein interactions. In *Proceedings of the Workshop on Building and Evaluating Resources for Biomedical Text Mining at LREC*.

E. Briscoe, J. Carroll, and R. Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL Interactive Presentation Sessions*.

R. Bunescu and M. Paşca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of EACL*.

L. Chen, H. Liu, and C. Friedman. 2005. Gene name ambiguity of eukaryotic nomenclatures. *Bioinformatics*, 21(2):248–256.

S. Clark and J. R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4).

M-C de Marneffe, B. MacCartney, and C. D. Manning. 2006. Generating typed dependency parses from phrase structure. In *Proceedings of LREC*.

G. Erkan, A. Ozgur, and D. R. Radev. 2007. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In *Proceedings of the Joint Conference of EMNLP and CoNLL*.

G. Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305.

C. Grover, M. Lapata, and A. Ascarides. 2003. A comparison of parsing technologies for the biomedical domain. *Natural Language Engineering*, 1(1):1–38.

J. Hakenberg, C. Plake, R. Leaman, M. Schroeder, and G. Gonzalez. 2008. Inter-species normalization of gene mentions with GNAT. *Bioinformatics*, 24(16).

T. Hara, Y. Miyao, and J. Tsujii. 2007. Evaluating impact of re-training a lexical disambiguation model on domain adaptation of an HPSG parser. In *Proceedings of the 10th International Conference on Parsing Technology*.

V. Hatzivassiloglou, PA Duboué, and A. Rzhetsky. 2001. Disambiguating proteins, genes, and RNA in text: a machine learning approach. *Bioinformatics*, 17(Suppl 1).

L. Hirschman, M. Krallinger, J. Wilbur, and A. Valencia, editors. 2007. *The BioCreative II - Critical Assessment for Information Extraction in Biology Challenge*, volume 9(Suppl 2). Genome Biology.

L. Hunter and K. B. Cohen. 2006. Biomedical language processing: what's beyond PubMed. *Molecular Cell*, 21(5):589–594.

N. Japkowicz. 2000. Learning from imbalanced data sets: a comparison of various strategies. In *Proceedings of AAAI Workshop on Learning from Imbalanced Data Sets*.

D. Klein and C. D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*.

R. Koeling, D. McCarthy, and J. Carroll. 2005. Domain-specific sense distributions and predominant sense acquisition. In *Proceedings of HLT/EMNLP*.

D. Lin. 1998. Dependency-based evaluation of MINIPAR. In *Proceedings of Workshop on the Evaluation of Parsing Systems*.

M. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. Miwa, R. Satre, Y. Miyao, T. Ohta, and J. Tsujii. 2008. Combining multiple layers of syntactic information for protein-protein interaction extraction. In *Proceedings of SMBM*.

Y. Miyao and J. Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1).

Y. Miyao, R. Sætre, K. Sagae, T. Matsuzaki, and J. Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of ACL-08: HLT*.

A. A. Morgan and L. Hirschman. 2007. Overview of BioCreAtIvE II gene normalisation. In *Proceedings of the BioCreAtIvE II Workshop*, Madrid.

A. A. Morgan, Z. Lu, X. Wang, A. M. Cohen, J. Fluck, P. Ruch, A. Divoli, K. Fundel, R. Leaman, J. Hakenberg, C. Sun, H. Liu, R. Torres, M. Krauthammer, W. W. Lau, H. Liu, C. Hsu, M. Schuemie, K. B. Cohen, and L. Hirschman. 2008. Overview of BioCreAtIvE II gene normalization. *Genome Biology*, 9(Suppl 2).

A. Moschitti. 2006. Making tree kernels practical for natural language learning. In *Proceedings of EACL*.

E. W. Noreen. 1989. *Computer Intensive Methods for Testing Hypothesis*. John Wiley & Sons.

C. Pollard and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.

A. S. Schwartz and M. A. Hearst. 2003. Identifying abbreviation definitions in biomedical text. In *Proceedings of the Pacific Symposium on Biocomputing*.

B. Settles. 2005. ABNER: An open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(14):3191–3192.

M. Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

Y. Tateisi, A. Yakushiji, T. Ohta, and J. Tsujii. 2005. Syntax annotation for the GENIA corpus. In *Proceedings of IJCNLP*.

X. Wang and C. Grover. 2008. Learning the species of biomedical named entities from annotated corpora. In *Proceedings of LREC*.

Y. Yang and J. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of ICML*.

A. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of COLING*.

Domain adaptive bootstrapping for named entity recognition

Dan Wu[1], **Wee Sun Lee**[2], **Nan Ye**[2]
[1]Singapore MIT Alliance
[2]Department of Computer Science
National University of Singapore
{dwu@,leews@comp,g0701171@}nus.edu.sg

Hai Leong Chieu
DSO National Laboratories
chaileon@dso.org.sg

Abstract

Bootstrapping is the process of improving the performance of a trained classifier by iteratively adding data that is labeled by the classifier itself to the training set, and retraining the classifier. It is often used in situations where labeled training data is scarce but unlabeled data is abundant. In this paper, we consider the problem of domain adaptation: the situation where training data may not be scarce, but belongs to a different domain from the target application domain. As the distribution of unlabeled data is different from the training data, standard bootstrapping often has difficulty selecting informative data to add to the training set. We propose an effective domain adaptive bootstrapping algorithm that selects unlabeled target domain data that are *informative about the target domain* and *easy to automatically label correctly*. We call these instances *bridges*, as they are used to bridge the source domain to the target domain. We show that the method outperforms supervised, transductive and bootstrapping algorithms on the named entity recognition task.

1 Introduction

Most recent researches on natural language processing (NLP) problems are based on machine learning algorithms. High performance can often be achieved if the system is trained and tested on data from the same domain. However, the performance of NLP systems often degrades badly when the test data is drawn from a source that is different from the labeled data used to train the system. For named entity recognition (NER), for example, Ciaramita and Altun (2005) reported that a system trained on a labeled Reuters corpus achieved an F-measure of 91% on a Reuters test set, but only 64% on a Wall Street Journal test set.

The task of adapting a system trained on one domain (called the *source* domain) to a new domain (called the *target* domain) is called domain adaptation. In domain adaptation, it is generally assumed that we have labeled data in the source domain while labeled data may or may not be available in the target domain. Previous work in domain adaptation can be classified into two categories: [S+T+], where a small, labeled target domain data is available, e.g. (Blitzer et al., 2006; Jiang and Zhai, 2007; Daumé III, 2007; Finkel and Manning, 2009), or [S+T-], where no labeled target domain data is available, e.g. (Blitzer et al., 2006; Jiang and Zhai, 2007). In both cases, and especially for [S+T-], domain adaptation can leverage on large amounts of unlabeled data in the target domain. In practice, it is often unreasonable to expect labeled data for every new domain that we come across, such as blogs, emails, a different newspaper agency, or simply articles from a different topic or period in time. Thus although [S+T+] is easier to handle, [S+T-] is of higher practical importance.

In this paper, we propose a domain adaptive bootstrapping (DAB) approach to tackle the domain adaptation problem under the setting [S+T-]. Bootstrapping is an iterative process that uses a trained classifier to label and select unlabeled instances to add to the training set for retraining the classifier. It is often used when labeled training data is scarce but unlabeled data is abundant. In contrast, for domain adaptation problems, we may have a lot of training data but the target application domain has a different data distribution. Standard bootstrapping usually selects instances that are most confidently labeled from the unlabeled data. In domain adaptation situations, usually the most confidently labeled instances are the ones that are most similar to the source domain in-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1523–1532,
Singapore, 6-7 August 2009.

stances - these instances tend to contain very little information about the target domain. For domain adaptive bootstrapping, we propose a selection criterion that selects instances that are informative and easy to automatically label correctly. In addition, we propose a criterion for stopping the process of bootstrapping before it adds uninformative and incorrectly labeled instances that can reduce performance.

Our approach leverages on instances in the target domain called *bridges*. These instances contain domain-independent features, as well as features specific to the target domain. As they contain domain-independent features, they can be classified correctly by classifiers trained on the source domain labeled data. We argue that these instances act as a bridge between the source and the target domain. We show that, on the NER task, DAB outperforms supervised, transductive and standard bootstrapping algorithms, as well as a bootstrapping variant, called *balanced* bootstrapping (Jiang and Zhai, 2007), that has recently been proposed for domain adaptation.

2 Related work

One general class of approaches to domain adaptation is to consider that the instances from the source and the target domain are drawn from different distributions. Bickel et al. (Bickel et al., 2007) discriminatively learns a scaling factor for source domain training data, so as to adapt the source domain data distribution to resemble the target domain data distribution, under the [S+T-] setting. Daume III and Marcu (Daumé III and Marcu, 2006) considers that the data distribution is a mixture distribution over general, source domain and target domain data. They learn the underlying mixture distribution using the conditional expectation maximization algorithm, under the [S+T+] setting. Jiang and Zhai (2007) proposed an instance re-weighting framework that handles both the [S+T+] and [S+T-] settings. For [S+T-], the resulting algorithm is a *balanced* bootstrapping algorithm, which was shown to outperform the *standard* bootstrapping algorithm. In this paper, we assume the [S+T-] settings, and we show that the approach proposed in this paper, domain adaptive bootstrapping (DAB), outperforms the balanced bootstrapping algorithm on NER.

Another class of approaches to domain adaptation is feature-based. Daume III (Daumé III, 2007) divided features into three classes: domain-independent features, source-domain features and target-domain features. He assumed the existence of training data in the target-domain (under the setting [S+T+]), so that the three classes of features can be jointly trained using source and target domain labeled data. This cannot be done in the setting [S+T-], where no training data is available in the target domain. Using a different approach, Blitzer et al. (2006) induces correspondences between feature spaces in different domains, by detecting pivot features. Pivot features are features that occur frequently and behave similarly in different domains. Pivot features are used to put domain-specific features in correspondence. In this paper, instead of pivot features, we attempt to leverage on pivot instances that we call *bridges*, which are instances that bridge the source and target domain. This will be illustrated in Section 3.

It is generally recognized that adding informative and correctly labeled instances is more useful for learning. Active learning queries the user for labels of most informative or relevant instances. Active learning, which has been applied to the problem of NER in (Shen et al., 2004), is used in situations where a large amount of unlabeled data exists and data labeling is expensive. It has also been applied to the problem of domain adaptation for word sense disambiguation in (Chan and Ng, 2007). However, active learning requires human intervention. Here, we want to achieve the same goal without human intervention.

3 Bootstrapping for domain adaptation

We first define the notations used for domain adaptation in the [S+T-] setting. A set of training data $D_S = \{x_i, y_i\}_{1 \leq i \leq |D_S|}$ is given in the source domain, where the notation $|X|$ denotes the size of a set X. Each instance x_i in D_S has been manually annotated with a label, y_i, from a given set of labels Y. The objective of domain adaptation is to label a set of unlabeled data, $D_T = \{x_i\}_{1 \leq i \leq |D_T|}$ with labels from Y. A machine learning algorithm will take a labeled data set (for e.g. D_S) and outputs a *classifier*, which can then be used to classify unlabeled data, i.e. assign labels to unlabeled instances.

A special class of machine learning algorithms, called *transductive* learning algorithms, is able to take the unlabeled data D_T into account during the learning process (see e.g. (Joachims, 1999)).

However, such algorithms do not take into account the shift in domain of the test data. Jiang and Zhai (2007) recently proposed an instance re-weighting framework to take domain shift into account. For [S+T-], the resulting algorithm is a *balanced* bootstrapping algorithm, which we describe below.

3.1 Standard and balanced bootstrapping

We define a general bootstrapping algorithm in Algorithm 1. The algorithm can be applied to any machine learning algorithm that allows training instances to be weighted, and that gives confidence scores for the labels when used to classify test data. The bootstrapping procedure iteratively improves the performance of a classifier SC_t over a number of iterations. In Algorithm 1, we have left a number of parameters unspecified. These parameters are (1) the *selection-criterion* for instances to be added to the training data, (2) the *termination-criterion* for the bootstrapping process, and (3) the weights (w^S, w^T) given to the labeled and bootstrapped training sets.

Standard bootstrapping: (Jiang and Zhai, 2007) the *selection-criterion* is based on selecting the top k most-confidently labeled instances in R_t. The weight w_t^S is equal to w_t^T. The value of k is a parameter for the bootstrapping algorithm.

Balanced bootstrapping: (Jiang and Zhai, 2007) the *selection-criterion* is still based on selecting the top k most-confidently labeled instances in R_t. Balanced bootstrapping was formulated for domain adaptation, and hence they set the weights to satisfy the ratio $\frac{w_t^S}{w_t^T} = \frac{|T_t|}{|D_S|}$. This allows the small amount of target data added, T_t, to have an equal weight to the large source domain training set D_S.

In this paper, we formulate a selection-criterion and a termination-criterion which are better than those used in standard and balanced bootstrapping. Regarding the selection-criterion, standard and balanced bootstrapping both select instances which are confidently labeled by SC_t to be used for training SC_{t+1}, in the hope of avoiding using wrongly labeled data in bootstrapping. However, instances that are already confidently labeled by SC_t may not contain sufficient information which is not in D_S, and using them to train SC_{t+1} may result in SC_{t+1} performing similarly to SC_t. This motivates us to select samples which are both *informative* and *easy to automatically label correctly*. Regarding the *termination-criterion*, which is not mentioned in the paper (Jiang and Zhai, 2007), we assume that bootstrapping is simply run for either a single iteration, or a small and fixed number of iterations. However, it is known that such simple criterion may result in stopping too early or too late, leading to sub-optimal performance. We propose a more effective termination-criterion here.

Algorithm 1 Bootstrapping algorithm

Input: labeled data D_S, test data D_T and a machine learning algorithm.
Output: the predicted labels of the set D_T.
Set $T_0 = \emptyset$, $R_0 = D_T$, and $t = 0$
Repeat

1. learn a classifier SC_t with (D_S, T_t) with weights (w_t^S, w_t^T)
2. label the set R_t with SC_t
3. select $S_t \subseteq R_t$ based on *selection-criterion*
4. $T_{t+1} = T_t \cup S_t$, and $R_{t+1} = R_t \setminus S_t$.

Until *termination-criterion*
Output the predicted labels of D_T by SC_t.

3.2 Domain adaptive bootstrapping (DAB)

Our selection-criterion relies on the observation that in domain adaptation, instances (from the source or the target domain) can be divided into three types according to their information content: *generalists* are instances that contain only domain-independent information and are present in all domains; *specialists* are instances containing only domain-specific information and are present only in their respective domains; *bridges* are instances containing both domain-independent and domain-specific information, also present only in their respective domains but are useful as a "bridge" between the source and the target domains.

The implication of the above observation is that when choosing unlabeled target domain data for bootstrapping, we should exploit the *bridges*, because the *generalists* are not likely to contain much information not in D_S due to their domain-independence, and the *specialists* are difficult to be labeled correctly due to their domain-specificity. In contrast, the *bridges* are informative and easier to label correctly. Choosing confidently classified instances for bootstrapping, as in standard bootstrapping and balanced bootstrapping, is simple, but results in choosing mostly *generalists*, and is too conservative. We design a scoring function

on instances, which has high value when the instance is informative and sufficiently likely to be correctly labeled in order to identify correctly labeled *bridges*.

Intuitively, informativeness of an instance can be measured by the prediction results of the ideal classifier IS for the source domain and the ideal classifier IT for the target domain. If IS and IT are both probabilistic classifiers, IS should return a noninformative distribution while IT should return an informative one. The ideal classifier for the source domain is approximated with a *source classifier* SC trained on D_S, while the ideal classifier for the target domain is approximated by training a classifier, TC, on target domain instances labeled by the source classifier.

We also try to ensure that instances that are selected are correctly classified. As the label used is provided by the target classifier, we estimate the precision of the target classification. The final ranking function is constructed by combining this estimate with the informativeness of the instance.

We show the algorithm for the instance selection in Algorithm 2. The notations used follow those used in Algorithm 1. For simplicity, we assume that $w_t^S = w_t^T = 1$ for all t. We expect TC to be a reasonable classifier on D_T due to the presence of *generalists* and *bridges*. Note that the target classifier is constructed by randomly splitting D_T into two partitions, training a classifier on each partition and using the prediction of the trained classifier on the partition it is not trained on. This is because classifiers tend to fit the data that they have been trained on too well making the probability estimates on their training data unreliable. Also, a random partition is used to ensure that the data in each partition is representative of D_u.

3.3 The scoring function: $score(p^{(s)}, p^{(t)})$

The scoring function $score(p^{(s)}, p^{(t)})$ in Algorithm 2 is simply implemented as the product of two components: a measure of the informativeness and the probability that SC's label is correct. We show how the intuitive ideas (described above) behind these two components are formalized.

Informativeness of a distribution p on a set of discrete labels Y is measured by its entropy $h(p)$ defined by

$$h(p) = -\sum_{y \in Y} p(y) \log p(y).$$

Algorithm 2 Algorithm for selecting instances for bootstrapping at iteration t

Input: Labeled source domain data D_S, target domain training data T_t, remaining data R_t, the classifier SC_t trained on $D_S \cup T_t$, and a scoring function $score(p^{(s)}, p^{(t)})$

Output: k instances for bootstrapping.

1. Label R_t with SC_t, and to each instance $x_i \in R_t$, SC_t outputs a distribution $p_i^{(s)}(y_i)$ over its labels.
2. Randomly split R_t into two partitions, R_t^0 and R_t^1 with their labels assigned by SC_t.
3. Train each target classifier, TC_t^x with the data R_t^x, for $x = \{0, 1\}$.
4. Label $R_t^{(1-x)}$ with the classifier TC_t^x, which to each instance $x_i \in R_t$, outputs a distribution $p_i^{(t)}(y_i)$ over its labels.
5. Score each instance from $x_i \in R_t$ with the function $score(p_i^{(s)}, p_i^{(t)})$.
6. Select top k instances from R_t with the highest scores.

$h(p)$ is nonnegative; $h(p) = 0$ if and only if p has probability 1 on one of the labels; $h(p)$ attains its maximum value when the distribution p is uniform over all labels. Hence, an instance is classified with high confidence when the distribution over its labels has low entropy.

We measure the informativeness of an instance using $h(p^{(s)}) - h(p^{(t)})$, where $p^{(s)}$ and $p^{(t)}$ are as in Algorithm 2. We argue that a larger value of this expression implies that the instance is more likely to be a *bridge* instance. This expression has a high value when the source classifier is uncertain, and the target classifier is certain. Uncertain classification by the source classifier indicates that the instance is unlikely to be a *generalist*. Moreover, if the target classifier is certain on x_i, it means that instances similar to the instance x_i are consistently labeled with the same label by the source classifier SC_t, indicating that it is likely to be a *bridge* instance.

The probability that TC's label is correct cannot be estimated directly because we do not have labeled target domain data. Instead, we use the source domain to give an estimate. We do this with a simple pre-processing step: we split the data D_S into two partitions of equal size, train a classifier on each partition, and test each classifier on the

other partition. We then measure the resulting accuracy given each label:

$$\rho(y) = \frac{\text{\# correctly labeled instances of label } y}{\text{\# total instances of label } y}.$$

Summarizing the above discussion, the scoring function is as shown below.

$$score(p^{(s)}, p^{(t)}) = \rho(y^*) \left[h(p^{(s)}) - h(p^{(t)})\right],$$
$$\text{where } y^* = \arg\max_{y \in Y} p^{(s)}(y)$$

The scoring function has a high value when the information content of the example is high and the label has high precision.

3.4 The termination criterion

Intuitively, our algorithm terminates when there are not enough informative instances. Formally, we define the termination criterion as follows: we terminate the bootstrapping process when, there exists an instance x_i in the top k instances satisfying the following condition:

1. $h(p_i^{(s)}) < h(p_i^{(t)})$, or
2. $\max_{y \in Y} p_i^{(s)}(y) > \max_{y \in Y} p_i^{(t)}(y)$

The second case is used to check for instances where the classifier SC_t is more confident than the target classifiers TC_t^x, on their respective predicted labels. This shows that the instance x_i is more of a *generalist* than a *bridge*.

4 NER task and implementation

The algorithm described in Section 3 is not specific to any particular application. In this paper, we apply it to the problem of named entity recognition (NER). In this section, we describe the NER classifier and the features used in our experiments.

4.1 NER features

We used the features generated by the CRF package (Finkel et al., 2005). These features include the word string feature, the case feature for the current word, the context words for the current word and their cases, the presence in dictionaries for the current word, the position of the current word in the sentence, prefix and suffix of the current word as well as the case information of the multiple occurrences of the current word. We use the same set of features for all classifiers used in the bootstrapping process, and for all baselines used in the experimental section.

4.2 Machine learning algorithms

A base machine learning algorithm is required in bootstrapping approaches. We describe the two machine learning algorithms used in this paper. We chose these algorithms for their good performance on the NER task.

Maximum entropy classification (MaxEnt): The MaxEnt approach, or logistic regression, is one of the most competitive methods for named entity recognition (Tjong and Meulder, 2003). MaxEnt is a discriminative method that learns a distribution, $p(y_i|x_i)$, over the labels, y_i, given the vector of features, x_i. We used the implementation of MaxEnt classifier described in (Manning and Klein, 2003). For NER, each instance represents a single word token within a sentence, with the feature vector x_i derived from the sentence as described in the previous section. MaxEnt is not designed for sequence classification. To deal with sequences, each name-class (e.g. PERSON) is divided into sub-classes: first token (e.g. PERSON-begin), unique token (e.g. PERSON-unique), or subsequent tokens (e.g. PERSON-continue) in the name-class. To ensure that the results returned by MaxEnt is coherent, we define deterministic transition probabilities that disallow transitions such as one from PERSON-begin to LOCATION-continue. A Viterbi parse is used to find the *valid* sequence of name-classes with the highest probability.

Support vector machines (SVM): The basic idea behind SVM for binary classification problems is to consider the data points in their feature space, and to separate the two classes with a hyper-plane, by maximizing the shortest distance between the data points and the hyper-plane. If there exists no hyperplane that can split the two labels, the soft margin version of SVM will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples (Joachims, 2002). We used the SVM^{light} package for our experiments (Joachims, 2002). For the multi-label NER classification with N classes, we learn N SVM classifiers, and use a softmax function to obtain the distribution. Formally, denoting by $s(y)$ the confidence returned by the classifier for each label $y \in Y$, the probability of the label y_i is given by

$$p(y_i|x_i) = \frac{\exp(s(y_i))}{\sum_{y \in Y} \exp(s(y))}$$

Similarly to MaxEnt, we subdivide name-classes into begin, continue, and unique sub-classes, and use a Viterbi parse for the sequence of highest probability. The SVM^{light} package also implements a transductive version of the SVM algorithm. We also compare our approach with the transductive SVM (Joachims, 1999) in our experimental results.

5 Experimental results

In this paper, we use the annotated data provided by the Automatic Content Extraction (ACE) program. The ACE data set is annotated for an Entity Detection task, and the annotation consists of the labeling of entity names (e.g. *Powell*) and mentions for each entity (e.g. pronouns such as *he*). In this paper, we are interested in the problem of recognition of the proper names (the named entity recognition task), and hence use only entities labeled with the type *NAM* (LDC, 2005). Entities are classified into seven types: *Person* entities are humans mentioned in a document; *Organization* entities are limited to established associations of people; *Geo-political* entities are geographical areas defined by political and/or social groups; *Location* entities are geographical items like landmasses and bodies of water; *Facility* entities refer to buildings and real estate improvements; *Vehicle* entities are devices used for transportation; and *Weapon* entities are devices used for harming or destruction.

We compare performances of a few algorithms: MaxEnt classifier (MaxEnt); MaxEnt classifier with standard bootstrapping (MaxEnt-SB); balanced bootstrapping based on MaxEnt classifier (MaxEnt-BB); MaxEnt with DAB (MaxEnt-DAB); SVM classifier (SVM); transductive SVM classifier (SVM-Trans); and DAB based on SVM classifier (SVM-DAB). No regularization is used for MaxEnt classifiers. SVM classifiers use a value of 10 for parameter C (trade-off between training error and margin). Bootstrapping based algorithms are run for 30 iterations and 100 instances are selected in every iteration.

The evaluation measure used is the F-measure. F-measure is the harmonic mean of precision and recall, and is commonly used to evaluate NER systems. We use the scorer for CONLL 2003 shared task (Tjong and Meulder, 2003) where the F-measure is computed by averaging F-measures for name-classes, weighted by the number of occurrences.

Code	Source	Num docs
NW	Newswire	81
BC	Broadcast conversation	52
WL	Weblog	114
CTS	Conversational Telephone Speech	34

Table 1: The sources, and the number of documents in each source, in the ACE 2005 data set.

5.1 Cross-source transfer

The ACE 2005 data set consists of articles drawn from a variety of sources. We use the four categories shown in Table 1. Each category is considered to be a domain, and we consider each pair of categories as the source and the target domain in turn.

Figure 1 compares the performance of MaxEnt-SB, MaxEnt-BB and MaxEnt-DAB over multiple iterations. Figure 2 compares the performance of SVM, SVM-Trans and SVM-DAB. Each line in the figures represents the average F-measure across all the domains over many iterations. When the termination condition is met for one domain, its F-measure remains at the value of the final iteration.

Despite a large number of iterations, both standard and balanced bootstrapping fail to improve performance. Supervised learning performance on each domain is shown in Table 3 (by 2-fold cross-validation with random ordering) as a reference. In Table 5, we compare the F-measures obtained by different algorithms at the last iteration they were run. We will discuss more on this in Section 5.3.

5.2 Cross-topic transfer

This data set is constructed from 175 articles from the ACE 2005 corpus. The data set is used to evaluate transfer across topics. We manually classify the articles into 4 categories: military operations (MO), political relationship or politicians (POL), terrorism-related (TER), and those which are not in the above categories (OTH). A detailed breakdown of the number of documents in the each topic is given in Table 2.

Supervised learning performance on each domain is shown in Table 4 (by 2-fold cross-validation with random ordering) as a reference. Experimental results on cross-topic evaluation are shown in Table 6. Figure 3 compares the performance of MaxEnt-SB, MaxEnt-BB and MaxEnt-

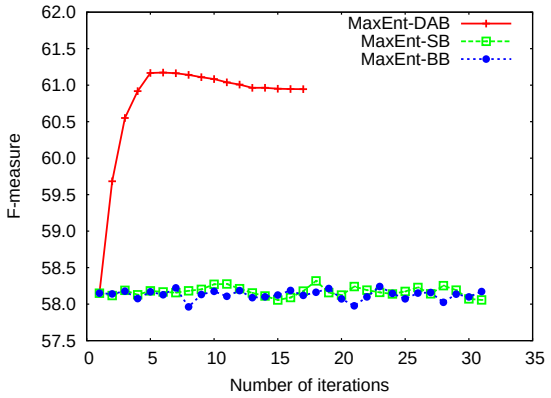


Figure 1: Average performance on the cross-source transfer using MaxEnt classifier.

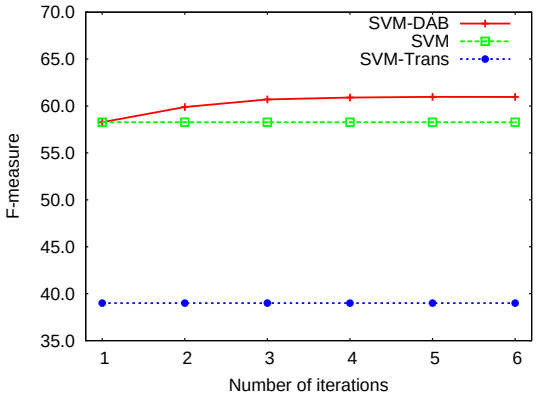


Figure 2: Average performance on the cross-source transfer using SVM classifier.

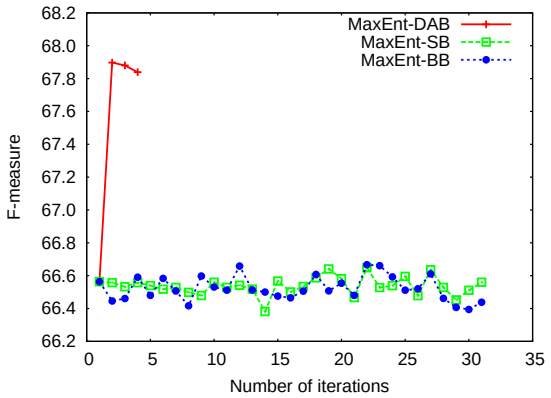


Figure 3: Average performance on the cross-topic transfer using MaxEnt classifier.

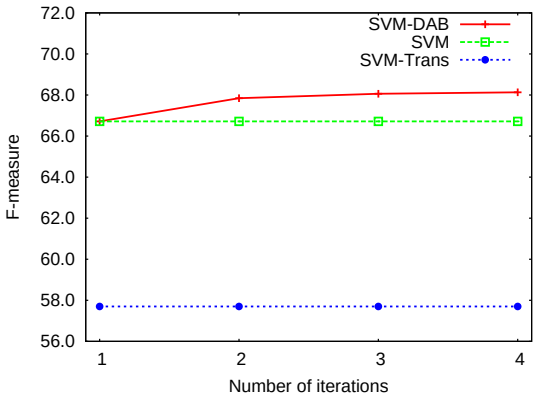


Figure 4: Average performance on the cross-topic transfer using SVM classifier.

Topic	Topic description	# docs
MO	Military operations	92
POL	Political relationships	40
TER	Terrorist-related	28
OTH	None of the above	15

Table 2: The topics, their descriptions, and the number of training and test documents in each topic.

Domain	MaxEnt	SVM
NW	82.47	82.32
BC	78.21	77.91
WL	71.41	71.84
CTS	93.90	94.01

Table 3: F-measure of supervised learning on the cross-source target domains.

DAB over multiple iterations. Figure 4 compares the performance of SVM, SVM-Trans and SVM-DAB. Similar to cross-source transfer, standard and balanced bootstrapping perform badly. This will be discussed in Section 5.3.

Domain	MaxEnt	SVM
MO	80.52	80.6
POL	77.99	79.05
TER	81.74	82.12
OTH	71.33	72.08

Table 4: F-measure of supervised learning on the cross-topic target domains.

5.3 Discussion

We show in our experiments that DAB outperforms standard and balanced bootstrapping, as well as the transductive SVM. We have also shown DAB to be robust across two state-of-the-art classifiers, MaxEnt and SVM. Balanced bootstrapping has been shown to be more effective for domain adaptation than standard bootstrapping (Jiang and Zhai, 2007) for named entity classification on a subset of the dataset used here. In contrast, we found that both methods perform poorly on domain adaptation for NER. In named entity classification, the names have already been segmented out and only need to be classified with the appropriate class. However, for NER, the names also

Train	Test	MaxEnt	MaxEnt-SB	MaxEnt-BB	MaxEnt-DAB	SVM	SVM-Trans	SVM-DAB
BC	CTS	74.26	74.19	74.16	**81.03**	72.47	43.27	75.43
BC	NW	64.81	64.76	64.80	**66.20**	64.08	43.01	64.39
BC	WL	47.81	47.80	47.76	**49.52**	47.98	36.58	47.93
CTS	BC	46.19	46.12	46.40	**54.62**	46.02	40.44	49.64
CTS	NW	54.25	54.15	54.26	53.07	55.63	23.61	**58.99**
CTS	WL	40.42	40.43	40.72	41.27	39.96	29.05	**42.04**
NW	BC	59.90	59.83	59.80	**60.55**	59.89	45.71	58.42
NW	CTS	66.64	66.48	66.59	66.73	68.28	28.80	**73.47**
NW	WL	52.52	52.53	52.47	**53.44**	52.19	36.39	52.30
WL	BC	58.58	**58.79**	58.65	56.00	58.43	52.64	58.64
WL	CTS	64.63	63.89	64.50	80.45	65.96	45.04	**81.04**
WL	NW	67.79	67.72	67.92	68.46	68.38	43.40	**69.33**
Average		58.15	58.06	58.17	60.95	58.27	39.00	**60.97**

Table 5: F-measure of the cross-source transfer.

Train	Test	MaxEnt	MaxEnt-SB	MaxEnt-BB	MaxEnt-DAB	SVM	SVM-Trans	SVM-DAB
MO	OTH	81.70	81.48	81.57	**81.95**	81.78	75.68	81.94
MO	POL	73.21	73.11	73.28	**74.97**	72.56	58.13	72.66
MO	TER	68.13	68.07	68.24	**69.89**	69.40	65.02	69.38
OTH	MO	63.30	63.80	63.94	63.91	64.18	61.03	**65.45**
OTH	POL	67.96	68.05	67.86	69.13	68.29	56.50	**70.67**
OTH	TER	45.34	44.82	45.30	51.06	45.71	48.77	**52.87**
POL	MO	62.14	62.12	61.95	61.94	61.98	51.67	**62.32**
POL	OTH	77.91	77.72	77.79	76.58	78.11	65.71	**78.13**
POL	TER	66.55	66.38	66.08	66.38	66.44	51.29	**67.24**
TER	MO	58.35	**58.62**	58.02	57.29	58.30	49.80	58.14
TER	OTH	66.83	67.61	66.83	**68.97**	66.28	58.25	68.12
TER	POL	67.34	66.94	67.16	**72.00**	67.54	50.55	70.65
Average		66.56	66.56	66.50	67.84	66.71	57.70	**68.13**

Table 6: F-measure of the cross-topic transfer.

need to be separated from not-a-name instances. We find that the addition of not-a-name instances changes the problem - the not-a-names form most of the instances classified with high confidence. As a result, we find that both standard and balanced bootstrapping fail to improve performance: the selection of the most confident instances no longer provide sufficient new information to improve performance.

We also find that transductive SVM performs poorly on this task. This is because it assumes that the unlabeled data comes from the same distribution as the labeled data. In general, applying semi-supervised learning methods directly to [S+T-] type domain adaptation problems do not work and appropriate modifications need to be made to the methods.

The ACE 2005 data set also contains a set of ariticles from the *broadcast news* (BN) source which is written entirely in lower case. This makes NER much more difficult. However, when BN is the source domain, the capitalization information can be discovered by DAB. Figures 5 and 6 show the average performance when BN is used as the source domain and all other domains in Table 1 as the target domains.

The source domain classifier tends to have high precision and low recall, DAB results in an increase in recall, with a small decrease in precision.

Testing the significance of the F-measure is not trivial because the named entities wrongly labeled by two classifiers are not directly comparable. We tested the labeling disagreements instead, using a McNemar paired test. The significance test is performed on the improvement of MaxEnt-DAB over MaxEnt and SVM-DAB over SVM. In most of the domains for the cross-source transfer, the improvements are significant at a significance level of 0.05, using MaxEnt classifier. The exceptional train-test pairs are NW-WL and WL-BC. In the case of WL-BC, this means the slight decrement in performance is not statistically significant. Similar result is achieved for the cross-source transfer using SVM classifier. In the cross-topic transfer, the source domain and the target domain are not very different. When we have a large amount of training data and little testing data, the gain of DAB can be not statistically significant, as in the case when we train with MO and POL domains.

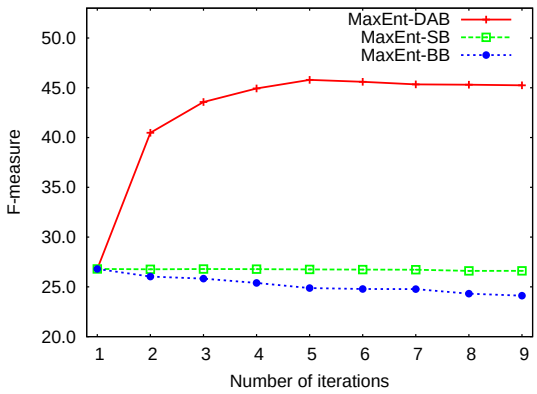


Figure 5: Performance on recovering capitalization using MaxEnt classifier.

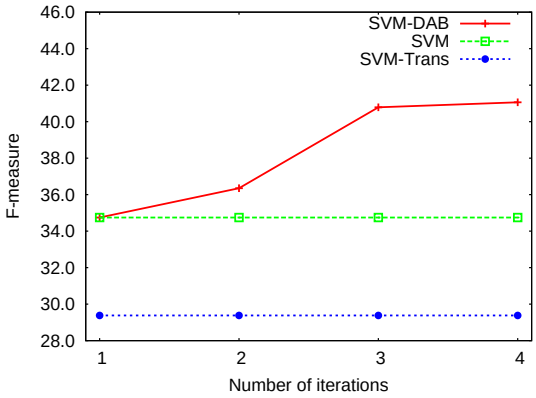


Figure 6: Performance on recovering capitalization using SVM classifier.

6 Conclusion

We proposed a bootstrapping approach for domain adaptation, and we applied it to the named entity recognition task. Our approach leverages on instances that serve as *bridges* between the source and target domain. Empirically, our method outperforms baseline approaches including supervised, transductive and standard bootstrapping approaches. It also outperforms balanced bootstrapping, an approach designed for domain adaptation (Jiang and Zhai, 2007).

References

Steffen Bickel, Michael Brückner, and Tobias Scheffer. 2007. Discriminative learning for differing training and test distributions. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 81–88, New York, NY, USA. ACM Press.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia.

Yee Seng Chan and Hwee Tou Ng. 2007. Domain adaptation with active learning for word sense disambiguation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 49–56, Prague, Czech Republic, June. Association for Computational Linguistics.

Massimiliano Ciaramita and Yasemin Altun. 2005. Named-entity recognition in novel domains with external lexical knowledge. In *Advances in Structured Learning for Text and Speech Processing Workshop*.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Conference of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic.

Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical bayesian domain adaptation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, New York City, USA. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA. Association for Computational Linguistics.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic, June. Association for Computational Linguistics.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

T. Joachims. 2002. *Learning to Classify Text Using Support Vector Machines – Methods, Theory, and Algorithms*. Kluwer/Springer.

Linguistic Data Consortium LDC. 2005. ACE (Automatic Content Extraction) English Annotation Guidelines for Entities.

Christopher Manning and Dan Klein. 2003. Optimization, maxent models, and conditional estimation without magic. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on*

Human Language Technology, pages 8–8, Morristown, NJ, USA. Association for Computational Linguistics.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 589–596, Barcelona, Spain, July.

Erik Tjong and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of Conference on Computational Natural Language Learning*.

Phrase Dependency Parsing for Opinion Mining

Yuanbin Wu, Qi Zhang, Xuanjing Huang, Lide Wu
Fudan University
School of Computer Science
{ybwu,qi_zhang,xjhuang,ldwu}@fudan.edu.cn

Abstract

In this paper, we present a novel approach for mining opinions from product reviews, where it converts opinion mining task to identify product features, expressions of opinions and relations between them. By taking advantage of the observation that a lot of product features are phrases, a concept of phrase dependency parsing is introduced, which extends traditional dependency parsing to phrase level. This concept is then implemented for extracting relations between product features and expressions of opinions. Experimental evaluations show that the mining task can benefit from phrase dependency parsing.

1 Introduction

As millions of users contribute rich information to the Internet everyday, an enormous number of product reviews are freely written in blog pages, Web forums and other consumer-generated mediums (CGMs). This vast richness of content becomes increasingly important information source for collecting and tracking customer opinions. Retrieving this information and analyzing this content are impossible tasks if they were to be manually done. However, advances in machine learning and natural language processing present us with a unique opportunity to automate the decoding of consumers' opinions from online reviews.

Previous works on mining opinions can be divided into two directions: sentiment classification and sentiment related information extraction. The former is a task of identifying positive and negative sentiments from a text which can be a passage, a sentence, a phrase and even a word (Somasundaran et al., 2008; Pang et al., 2002; Dave et al., 2003; Kim and Hovy, 2004; Takamura et al., 2005). The latter focuses on extracting the elements composing a sentiment text. The elements include source of opinions who expresses an opinion (Choi et al., 2005); target of opinions which is a receptor of an opinion (Popescu and Etzioni, 2005); opinion expression which delivers an opinion (Wilson et al., 2005b). Some researchers refer this information extraction task as opinion extraction or opinion mining. Comparing with the former one, opinion mining usually produces richer information.

In this paper, we define an opinion unit as a triple consisting of a product feature, an expression of opinion, and an emotional attitude(positive or negative). We use this definition as the basis for our opinion mining task. Since a product review may refer more than one product feature and express different opinions on each of them, the relation extraction is an important subtask of opinion mining. Consider the following sentences:

1. I highly [recommend]$^{(1)}$ the Canon SD500$^{(1)}$ to anybody looking for a compact camera that can take [good]$^{(2)}$ pictures$^{(2)}$.

2. This camera takes [amazing]$^{(3)}$ image qualities$^{(3)}$ and its size$^{(4)}$ [cannot be beat]$^{(4)}$.

The phrases underlined are the product features, marked with square brackets are opinion expressions. Product features and opinion expressions with identical superscript compose a relation. For the first sentence, an opinion relation exists between "*the Canon SD500*" and "*recommend*", but not between "*picture*" and "*recommend*". The example shows that more than one relation may appear in a sentence, and the correct relations are not simple Cartesian product of opinion expressions and product features.

Simple inspection of the data reveals that product features usually contain more than one word, such as "LCD screen", "image color", "Canon PowerShot SD500", and so on. An incomplete product feature will confuse the successive analysis. For example, in passage *"Image color is dis-*

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1533–1541,
Singapore, 6-7 August 2009.

appointed", the negative sentiment becomes obscure if only "*image*" or "*color*" is picked out.

Since a product feature could not be represented by a single word, dependency parsing might not be the best approach here unfortunately, which provides dependency relations only between words. Previous works on relation extraction usually use the head word to represent the whole phrase and extract features from the word level dependency tree. This solution is problematic because the information provided by the phrase itself can not be used by this kind of methods. And, experimental results show that relation extraction task can benefit from dependencies within a phrase.

To solve this issue, we introduce the concept of phrase dependency parsing and propose an approach to construct it. Phrase dependency parsing segments an input sentence into "phrases" and links segments with directed arcs. The parsing focuses on the "phrases" and the relations between them, rather than on the single words inside each phrase. Because phrase dependency parsing naturally divides the dependencies into local and global, a novel tree kernel method has also been proposed.

The remaining parts of this paper are organized as follows: In Section 2 we discuss our phrase dependency parsing and our approach. In Section 3, experiments are given to show the improvements. In Section 4, we present related work and Section 5 concludes the paper.

2 The Approach

Fig. 1 gives the architecture overview for our approach, which performs the opinion mining task in three main steps: (1) constructing phrase dependency tree from results of chunking and dependency parsing; (2) extracting candidate product features and candidate opinion expressions; (3) extracting relations between product features and opinion expressions.

2.1 Phrase Dependency Parsing

2.1.1 Overview of Dependency Grammar

Dependency grammar is a kind of syntactic theories presented by Lucien Tesnière(1959). In dependency grammar, structure is determined by the relation between a head and its dependents. In general, the dependent is a modifier or complement; the head plays a more important role in determining the behaviors of the pair. Therefore, criteria of how to establish dependency relations and how to distinguish the head and dependent in such relations is central problem for dependency grammar. Fig. 2(a) shows the dependency representation of an example sentence. The root of the sentence is "enjoyed". There are seven pairs of dependency relationships, depicted by seven arcs from heads to dependents.

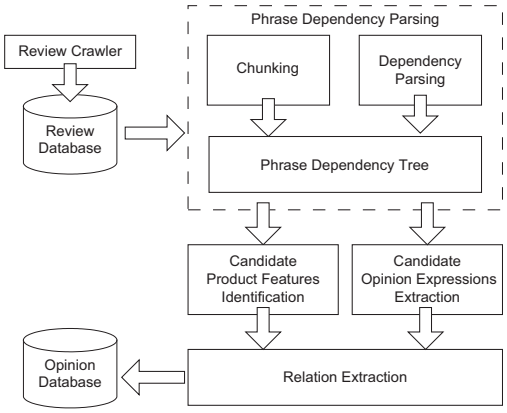


Figure 1: The architecture of our approach.

2.1.2 Phrase Dependency Parsing

Currently, the mainstream of dependency parsing is conducted on lexical elements: relations are built between single words. A major information loss of this word level dependency tree compared with constituent tree is that it doesn't explicitly provide local structures and syntactic categories (i.e. NP, VP labels) of phrases (Xia and Palmer, 2001). On the other hand, dependency tree provides connections between distant words, which are useful in extracting long distance relations. Therefore, compromising between the two, we extend the dependency tree node with phrases. That implies a noun phrase "Cannon SD500 PowerShot" can be a dependent that modifies a verb phrase head "really enjoy using" with relation type "dobj". The feasibility behind is that a phrase is a syntactic unit regardless of the length or syntactic category (Santorini and Kroch, 2007), and it is acceptable to substitute a single word by a phrase with same syntactic category in a sentence.

Formally, we define the dependency parsing with phrase nodes as *phrase dependency parsing*. A dependency relationship which is an asymmetric binary relationship holds between two phrases. One is called head, which is the central phrase in the relation. The other phrase is called dependent, which modifies the head. A label representing the

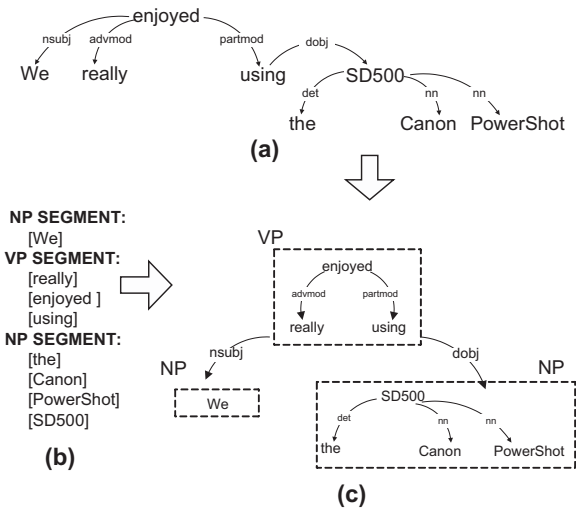


Figure 2: Example of Phrase Dependency Parsing.

relation type is assigned to each dependency relationship, such as subj (subject), obj (object), and so on. Fig.2(c) shows an example of phrase dependency parsing result.

By comparing the phrase dependency tree and the word level dependency tree in Fig.2, the former delivers a more succinct tree structure. Local words in same phrase are compacted into a single node. These words provide local syntactic and semantic effects which enrich the phrase they belong to. But they should have limited influences on the global tree topology, especially in applications which emphasis the whole tree structures, such as tree kernels. Pruning away local dependency relations by additional phrase structure information, phrase dependency parsing accelerates following processing of opinion relation extraction .

To construct phrase dependency tree, we propose a method which combines results from an existing shallow parser and a lexical dependency parser. A phrase dependency tree is defined as $\mathscr{T} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ is the set of phrases, $\mathscr{E}$ is the dependency relations among the phrases in $\mathscr{V}$ representing by direct edges. To reserve the word level dependencies inside a phrase, we define a nested structure for a phrase T_i in $\mathscr{V}$: $T_i = (V_i, E_i)$. $V_i = \{v_1, v_2, \cdots, v_m\}$ is the internal words, E_i is the internal dependency relations.

We conduct the phrase dependency parsing in this way: traverses word level dependency tree in preorder (visits root node first, then traverses the children recursively). When visits a node R, searches in its children and finds the node set D which are in the same phrase with R according to the shallow parsing result. Compacts D and R into a single node. Then traverses all the remaining children in the same way. The algorithm is shown in Alg. 1.

Algorithm 1 Pseudo-Code for constructing the phrase dependency tree

INPUT:
$T' = (V', E')$ a word level dependency tree
P = phrases
OUTPUT:
phrase dependency tree $\mathscr{T} = (\mathscr{V}, \mathscr{E})$ where
$\mathscr{V} = \{T_1(V_1, E_1), T_2(V_2, E_2), \cdots, T_n(V_n, E_n)\}$
Initialize:
$\mathscr{V} \leftarrow \{(\{v'\}, \{\}) | v' \in V'\}$
$\mathscr{E} \leftarrow \{(T_i, T_j) | (v'_i, v'_j) \in E', v'_i \in V_i, v'_j \in V_j\}$
$R = (V_r, E_r)$ root of $\mathscr{T}$
PhraseDPTree(R, P)

```
Find p_i ∈ P where word[R] ∈ p_i
for each S = (V_s, E_s), (R, S) ∈ 𝓔 do
    if word[S] ∈ p_i then
        V_r ← V_r ∪ v_s; v_s ∈ V_s
        E_r ← E_r ∪ (v_r, root[S]); v_r ∈ V_r
        𝓥 ← 𝓥 − S
        𝓔 ← 𝓔 + (R, l); ∀(S, l) ∈ 𝓔
        𝓔 ← 𝓔 − (R, S)
    end if
end for
for each (R, S) ∈ 𝓔 do
    PhraseDPTree(S,P)
end for
return (𝓥, 𝓔)
```

The output of the algorithm is still a tree, for we only cut edges which are compacted into a phrase, the connectivity is keeped. Note that there will be inevitable disagrees between shallow parser and lexical dependency parser, the algorithm implies that we simply follow the result of the latter one: the phrases from shallow parser will not appear in the final result if they cannot be found in the procedure.

Consider the following example:

"*We really enjoyed using the Canon PowerShot SD500.*"

Fig.2 shows the procedure of phrase dependency parsing. Fig.2(a) is the result of the lexical dependency parser. Shallow parsers result is shown in Fig.2(b). Chunk phrases "NP(We)", "VP(really enjoyed using)" and "NP(the Canon PowerShot SD500)" are nodes in the output phrase dependency tree. When visiting node "enjoyed" in Fig.2(a), the shallow parser tells that "really" and "using" which are children of "enjoy" are in the same phrase with their parent, then the three nodes are packed. The final phrase dependency parsing tree is shown in the Fig. 2(c).

2.2 Candidate Product Features and Opinion Expressions Extraction

In this work, we define that *product features* are products, product parts, properties of products, properties of parts, company names and related objects. For example,in consumer electronic domain, "Canon PowerShot", "image quality","camera", "laptop" are all product features.

From analyzing the labeled corpus, we observe that more than 98% of product features are in a single phrase, which is either noun phrase (NP) or verb phrase (VP). Based on it, all NPs and VPs are selected as candidate product features. While prepositional phrases (PPs) and adjectival phrases (ADJPs) are excluded. Although it can cover nearly all the true product features, the precision is relatively low. The large amount of noise candidates may confuse the relation extraction classifier. To shrink the size of candidate set, we introduce language model by an intuition that the more likely a phrase to be a product feature, the more closely it related to the product review. In practice, for a certain domain of product reviews, a language model is build on easily acquired unlabeled data. Each candidate NP or VP chunk in the output of shallow parser is scored by the model, and cut off if its score is less than a threshold.

Opinion expressions are spans of text that express a comment or attitude of the opinion holder, which are usually evaluative or subjective phrases. We also analyze the labeled corpus for opinion expressions and observe that many opinion expressions are used in multiple domains, which is identical with the conclusion presented by Kobayashi et al. (2007). They collected 5,550 opinion expressions from various sources . The coverage of the dictionary is high in multiple domains. Motivated by those observations, we use a dictionary which contains 8221 opinion expressions to select candidates (Wilson et al., 2005b). An assumption we use to filter candidate opinion expressions is that opinion expressions tend to appear closely with product features, which is also used to extract product features by Hu and Liu (2004). In our experiments, the tree distance between product feature and opinion expression in a relation should be less than 5 in the phrase dependency parsing tree.

2.3 Relation Extraction

This section describes our method on extracting relations between opinion expressions and product features using phrase dependency tree. Manually built patterns were used in previous works which have an obvious drawback that those patterns can hardly cover all possible situations. By taking advantage of the kernel methods which can search a feature space much larger than that could be represented by a feature extraction-based approach, we define a new tree kernel over phrase dependency trees and incorporate this kernel within an SVM to extract relations between opinion expressions and product features.

The potential relation set consists of the all combinations between candidate product features and candidate opinion expressions in a sentence. Given a phrase dependency parsing tree, we choose the subtree rooted at the lowest common parent(LCP) of opinion expression and product feature to represent the relation.

Dependency tree kernels has been proposed by (Culotta and Sorensen, 2004). Their kernel is defined on lexical dependency tree by the convolution of similarities between all possible subtrees. However, if the convolution containing too many irrelevant subtrees, over-fitting may occur and decreases the performance of the classifier. In phrase dependency tree, local words in a same phrase are compacted, therefore it provides a way to treat "local dependencies" and "global dependencies" differently (Fig. 3). As a consequence, these two kinds of dependencies will not disturb each other in measuring similarity. Later experiments prove the validity of this statement.

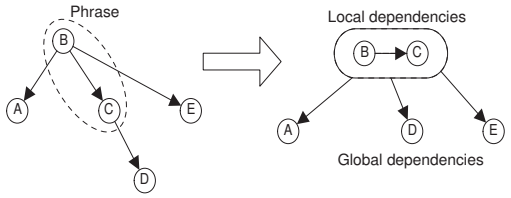


Figure 3: Example of "local dependencies" and "global dependencies".

We generalize the definition by (Culotta and Sorensen, 2004) to fit the phrase dependency tree. Use the symbols in Section 2.1.2, $\mathscr{T}^i$ and $\mathscr{T}^j$ are two trees with root R^i and R^j, $K(\mathscr{T}^i, \mathscr{T}^j)$ is the kernel function for them. Firstly, each tree node $T_k \in \mathscr{T}^i$ is augmented with a set of features F, and an instance of F for T_k is $F^k = \{f^k\}$. A match function $m(T_i, T_j)$ is defined on comparing a subset of nodes' features $M \subseteq F$. And in the same way, a similarity function $s(T_i, T_j)$ are de-

fined on $S \subseteq F$

$$m(T_i, T_j) = \begin{cases} 1 & if\ f_m^i = f_m^j \quad \forall f_m \in M \\ 0 & otherwise \end{cases} \quad (1)$$

and

$$s(T_i, T_j) = \sum_{f_s \in S} C(f_s^i, f_s^j) \quad (2)$$

where

$$C(f_s^i, f_s^j) = \begin{cases} 1 & if\ f_s^i = f_s^j \\ 0 & otherwise \end{cases} \quad (3)$$

For the given phrase dependency parsing trees, the kernel function $K(\mathscr{T}^i, \mathscr{T}^j)$ is defined as follow:

$$K(\mathscr{T}^i, \mathscr{T}^j) = \begin{cases} 0 & if\ m(R^i, R^j) = 0 \\ s(R^i, R^j) + K_{in}(R^i, R^j) & \\ + K_c(R^i.\mathbf{C}, R^j.\mathbf{C}) & otherwise \end{cases} \quad (4)$$

where $K_{in}(R^i, R^j)$ is a kernel function over $R^i = (V_r^i, E_r^i)$ and $R^j = (V_r^j, E_r^j)$'s internal phrase structures,

$$K_{in}(R^i, R^j) = K(R^i, R^j) \quad (5)$$

K_c is the kernel function over R^i and R^j's children. Denote **a** is a continuous subsequence of indices $a, a+1, \cdots a+l(\mathbf{a})$ for R^i's children where $l(\mathbf{a})$ is its length, $\mathbf{a}_s$ is the s-th element in **a**. And likewise **b** for R^j.

$$K_c(R^i.\mathbf{C}, R^j.\mathbf{C}) = \sum_{\mathbf{a},\mathbf{b},l(\mathbf{a})=l(\mathbf{b})} \lambda^{l(\mathbf{a})} K(R^i.[\mathbf{a}], R^j.[\mathbf{b}]) \times \prod_{s=1..l(\mathbf{a})} m(R^i.[\mathbf{a}_s], R^j.[\mathbf{b}_s]) \quad (6)$$

where the constant $0 < \lambda < 1$ normalizes the effects of children subsequences' length.

Compared with the definitions in (Culotta and Sorensen, 2004), we add term K_{in} to handle the internal nodes of a pharse, and make this extension still satisfy the kernel function requirements (composition of kernels is still a kernel (Joachims et al., 2001)). The consideration is that the local words should have limited effects on whole tree structures. So the kernel is defined on external children (K_c) and internal nodes (K_{in}) separately, as the result, the local words are not involved in subsequences of external children for K_c. After the kernel computing through training instances, support vector machine (SVM) is used for classification.

Table 1: Statistics for the annotated corpus

Category	# Products	# Sentences
Cell Phone	2	1100
Diaper	1	375
Digital Camera	4	1470
DVD Player	1	740
MP3 Player	3	3258

3 Experiments and Results

In this section, we describe the annotated corpus and experiment configurations including baseline methods and our results on in-domain and cross-domain.

3.1 Corpus

We conducted experiments with labeled corpus which are selected from Hu and Liu (2004), Jindal and Liu (2008) have built. Their documents are collected from Amazon.com and CNet.com, where products have a large number of reviews. They also manually labeled product features and polarity orientations. Our corpus is selected from them, which contains customer reviews of 11 products belong to 5 categories(Diaper, Cell Phone, Digital Camera, DVD Player, and MP3 Player). Table 1 gives the detail statistics.

Since we need to evaluate not only the product features but also the opinion expressions and relations between them, we asked two annotators to annotate them independently. The annotators started from identifying product features. Then for each product feature, they annotated the opinion expression which has relation with it. Finally, one annotator A_1 extracted 3595 relations, while the other annotator A_2 extracted 3745 relations, and 3217 cases of them matched. In order to measure the annotation quality, we use the following metric to measure the inter-annotator agreement, which is also used by Wiebe et al. (2005).

$$agr(a||b) = \frac{|A\ matches\ B|}{|A|}$$

Table 2: Results for extracting product features and opinion expressions

	P	R	F
Product Feature	42.8%	85.5%	57.0%
Opinion Expression	52.5%	75.2%	61.8%

Table 3: Features used in SVM-1: o denotes an opinion expression and t a product feature

1) Positions of o/t in sentence(start, end, other);
2) The distance between o and t (1, 2, 3, 4, other);
3) Whether o and t have direct dependency relation;
4) Whether o precedes t;
5) POS-Tags of o/t.

Table 4: Features used in SVM-PTree

Features for match function
1) The syntactic category of the tree node (e.g. NP, VP, PP, ADJP).
2) Whether it is an opinion expression node
3) Whether it is a product future node.
Features for similarity function
1) The syntactic category of the tree node (e.g. NP, VP, PP, ADJP).
2) POS-Tag of the head word of node's internal phrases.
3) The type of phrase dependency edge linking to node's parent.
4) Feature 2) for the node's parent
5) Feature 3) for the node's parent

where $agr(a||b)$ represents the inter-annotator agreement between annotator a and b, A and B are the sets of anchors annotated by annotators a and b. $agr(A_1||A_2)$ was 85.9% and $agr(A_2||A_1)$ was 89.5%. It indicates that the reliability of our annotated corpus is satisfactory.

3.2 Preprocessing Results

Results of extracting product features and opinion expressions are shown in Table 2. We use precision, recall and F-measure to evaluate performances. The candidate product features are extracted by the method described in Section 2.2, whose result is in the first row. 6760 of 24414 candidate product features remained after the filtering, which means we cut 72% of irrelevant candidates with a cost of 14.5%(1-85.5%) loss in true answers. Similar to the product feature extraction, the precision of extracting opinion expression is relatively low, while the recall is 75.2%. Since both product features and opinion expressions extractions are preprocessing steps, recall is more important.

3.3 Relation Extraction Experiments

3.3.1 Experiments Settings

In order to compare with state-of-the-art results, we also evaluated the following methods.

1. ***Adjacent*** method extracts relations between a product feature and its nearest opinion expression, which is also used in (Hu and Liu, 2004).

2. ***SVM-1***. To compare with tree kernel based approaches, we evaluated an SVM[1] result with a set of manually selected features(Table 3), which are also used in (Kobayashi et al., 2007).

3. ***SVM-2*** is designed to compare the effectiveness of cross-domain performances. The features used are simple bag of words and POS-Tags between opinion expressions and product features.

4. ***SVM-WTree*** uses head words of opinion expressions and product features in the word-level dependency tree, as the previous works in information extraction. Then conducts tree kernel proposed by Culotta and Sorensen (2004).

5. ***SVM-PTree*** denotes the results of our tree-kernel based SVM, which is described in the Section 2.3. Stanford parser (Klein and Manning, 2002) and Sundance (Riloff and Phillips, 2004) are used as lexical dependency parser and shallow parser. The features in match function and similarity function are shown in Table 4.

6. ***OERight*** is the result of SVM-PTree with correct opinion expressions.

7. ***PFRight*** is the result of SVM-PTree with correct product features.

Table 5 shows the performances of different relation extraction methods with in-domain data. For each domain, we conducted 5-fold cross validation. Table 6 shows the performances of the extraction methods on cross-domain data. We use the digital camera and cell phone domain as training set. The other domains are used as testing set.

[1]libsvm 2.88 is used in our experiments

Table 5: Results of different methods

	Cell Phone			MP3 Player			Digital Camera			DVD Player			Diaper		
Methods	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Adjacent	40.3%	60.5%	48.4%	26.5%	59.3%	36.7%	32.7%	59.1%	42.1%	31.8%	68.4%	**43.4%**	23.4%	78.8%	**36.1%**
SVM-1	69.5%	42.3%	52.6%	60.7%	30.6%	40.7%	61.4%	32.4%	42.4%	56.0%	27.6%	37.0%	29.3%	14.1%	19.0%
SVM-2	60.7%	19.7%	29.7%	63.6%	23.8%	34.6%	66.9%	23.3%	34.6%	66.7%	13.2%	22.0%	79.2%	22.4%	34.9%
SVM-WTree	52.6%	52.7%	52.6%	46.4%	43.8%	45.1%	49.1%	46.0%	47.5%	35.9%	32.0%	33.8%	36.6%	31.7%	34.0%
SVM-PTree	55.6%	57.2%	**56.4%**	51.7%	50.7%	**51.2%**	54.0%	49.9%	**51.9%**	37.1%	35.4%	36.2%	37.3%	30.5%	33.6%
OERight	66.7%	69.5%	68.1%	65.6%	65.9%	65.7%	64.3%	61.0%	62.6%	59.9%	63.9%	61.8%	55.8%	58.5%	57.1%
PFRight	62.8%	62.1%	62.4%	61.3%	56.8%	59.0%	59.7%	56.2%	57.9%	46.9%	46.6%	46.7%	58.5%	51.3%	53.4%

Table 6: Results for total performance with cross domain training data

	Diaper			DVD Player			MP3 Player		
Methods	P	R	F	P	R	F	P	R	F
Adjacent	23.4%	78.8%	36.1%	31.8%	68.4%	43.4%	26.5%	59.3%	36.7%
SVM-1	22.4%	30.6%	25.9%	52.8%	30.9%	39.0%	55.9%	36.8%	44.4%
SVM-2	71.9%	15.1%	25.0%	51.2%	13.2%	21.0%	63.1%	22.0%	32.6%
SVM-WTree	38.7%	52.4%	**44.5%**	30.7%	59.2%	40.4%	38.1%	47.2%	42.2%
SVM-PTree	37.3%	53.7%	44.0%	59.2%	48.3%	**46.3%**	43.0%	48.9%	**45.8%**

3.3.2 Results Discussion

Table 5 presents different methods' results in five domains. We observe that the three learning based methods(SVM-1, SVM-WTree, SVM-PTree) perform better than the Adjacent baseline in the first three domains. However, in other domains, directly adjacent method is better than the learning based methods. The main difference between the first three domains and the last two domains is the size of data(Table 1). It implies that the simple Adjacent method is also competent when the training set is small.

A further inspection into the result of first 3 domains, we can also conclude that: 1) Tree kernels(SVM-WTree and SVM-PTree) are better than Adjacent, SVM-1 and SVM-2 in all domains. It proofs that the dependency tree is important in the opinion relation extraction. The reason for that is a connection between an opinion and its target can be discovered with various syntactic structures. 2) The kernel defined on phrase dependency tree (SVM-PTree) outperforms kernel defined on word level dependency tree(SVM-WTree) by 4.8% in average. We believe the main reason is that phrase dependency tree provides a more succinct tree structure, and the separative treatment of local dependencies and global dependencies in kernel computation can indeed improve the performance of relation extraction.

To analysis the results of preprocessing steps' influences on the following relation extraction, we provide 2 additional experiments which the product features and opinion expressions are all correctly extracted respectively: OERight and PFRight. These two results show that given an exactly extraction of opinion expression and product feature, the results of opinion relation extraction will be much better. Further, opinion expressions are more influential which naturally means the opinion expressions are crucial in opinion relation extraction.

For evaluations on cross domain, the Adjacent method doesn't need training data, its results are the same as the in-domain experiments. Note in Table 3 and Table 4, we don't use domain related features in SVM-1, SVM-WTree, SVM-PTree, but SVM-2's features are domain dependent. Since the cross-domain training set is larger than the original one in Diaper and DVD domain, the models are trained more sufficiently. The final results on cross-domain are even better than in-domain experiments on SVM-1, SVM-WTree, and SVM-PTree with percentage of 4.6%, 8.6%, 10.3% in average. And the cross-domain training set is smaller than in-domain in MP3, but it also achieve competitive performance with the

in-domain. On the other hand, SVM-2's result decreased compared with the in-domain experiments because the test domain changed. At the same time, SVM-PTree outperforms other methods which is similar in in-domain experiments.

4 Related Work

Opinion mining has recently received considerable attention. Amount of works have been done on sentimental classification in different levels (Zhang et al., 2009; Somasundaran et al., 2008; Pang et al., 2002; Dave et al., 2003; Kim and Hovy, 2004; Takamura et al., 2005). While we focus on extracting product features, opinion expressions and mining relations in this paper.

Kobayashi et al. (2007) presented their work on extracting opinion units including: opinion holder, subject, aspect and evaluation. Subject and aspect belong to product features, while evaluation is the opinion expression in our work. They converted the task to two kinds of relation extraction tasks and proposed a machine learning-based method which combines contextual clues and statistical clues. Their experimental results showed that the model using contextual clues improved the performance. However since the contextual information in a domain is specific, the model got by their approach can not easily converted to other domains.

Choi et al. (2006) used an integer linear programming approach to jointly extract entities and relations in the context of opinion oriented information extraction. They identified expressions of opinions, sources of opinions and the linking relation that exists between them. The sources of opinions denote to the person or entity that holds the opinion.

Another area related to our work is opinion expressions identification (Wilson et al., 2005a; Breck et al., 2007). They worked on identifying the words and phrases that express opinions in text. According to Wiebe et al. (2005), there are two types of opinion expressions, direct subjective expressions and expressive subjective elements.

5 Conclusions

In this paper, we described our work on mining opinions from unstructured documents. We focused on extracting relations between product features and opinion expressions. The novelties of our work included: 1) we defined the phrase dependency parsing and proposed an approach to construct the phrase dependency trees; 2) we proposed a new tree kernel function to model the phrase dependency trees. Experimental results show that our approach improved the performances of the mining task.

6 Acknowledgement

This work was (partially) funded by Chinese NSF 60673038, Doctoral Fund of Ministry of Education of China 200802460066, and Shanghai Science and Technology Development Funds 08511500302. The authors would like to thank the reviewers for their useful comments.

References

Eric Breck, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In *Proceedings of IJCAI-2007*.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of HLT/EMNLP*.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings EMNLP*.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *In Proceedings of ACL 2004*.

Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *Proceedings of WWW 2003*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD 2004*.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of WSDM '08*.

Thorsten Joachims, Nello Cristianini, and John Shawe-Taylor. 2001. Composite kernels for hypertext categorisation. In *Proceedings of ICML '01*.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of Coling 2004*. COLING.

Dan Klein and Christopher D. Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *In Advances in Neural Information Processing Systems*.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting aspect-evaluation and aspect-of relations in opinion mining. In *Proceedings of EMNLP-CoNLL 2007*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proc. of EMNLP 2002*.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP*.

E. Riloff and W. Phillips. 2004. An introduction to the sundance and autoslog systems. In *University of Utah School of Computing Technical Report UUCS-04-015*.

Beatrice Santorini and Anthony Kroch. 2007. *The syntax of natural language: An online introduction using the Trees program*. http://www.ling.upenn.edu/ beatrice/syntax-textbook.

Swapna Somasundaran, Janyce Wiebe, and Josef Ruppenhofer. 2008. Discourse level opinion interpretation. In *Proceedings of COLING 2008*.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *Proceedings of ACL'05*.

L. Tesnière. 1959. *Eléments de syntaxe structurale*. Editions Klincksieck.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2/3).

Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005a. Opinionfinder: A system for subjectivity analysis. In *Demonstration Description in Conference on Empirical Methods in Natural Language Processing*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005b. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT-EMNLP*.

Fei Xia and Martha Palmer. 2001. Converting dependency structures to phrase structures. In *HLT '01: Proceedings of the first international conference on Human language technology research*.

Qi Zhang, Yuanbin Wu, Tao Li, Mitsunori Ogihara, Joseph Johnson, and Xuanjing Huang. 2009. Mining product reviews based on shallow dependency parsing. In *Proceedings of SIGIR 2009*.

Polynomial to Linear: Efficient Classification with Conjunctive Features

Naoki Yoshinaga
Institute of Industrial Science
University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo
ynaga@tkl.iis.u-tokyo.ac.jp

Masaru Kitsuregawa
Institute of Industrial Science
University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo
kitsure@tkl.iis.u-tokyo.ac.jp

Abstract

This paper proposes a method that speeds up a classifier trained with many conjunctive features: combinations of (primitive) features. The key idea is to precompute as partial results the weights of primitive feature vectors that appear frequently in the target NLP task. A trie compactly stores the primitive feature vectors with their weights, and it enables the classifier to find for a given feature vector its longest prefix feature vector whose weight has already been computed. Experimental results for a Japanese dependency parsing task show that our method speeded up the SVM and LLM classifiers of the parsers, which achieved accuracy of 90.84/90.71%, by a factor of 10.7/11.6.

1 Introduction

Deep and accurate text analysis based on discriminative models is not yet efficient enough as a component of real-time applications, and it is inadequate to process Web-scale corpora for knowledge acquisition (Pantel, 2007; Saeger et al., 2009) or semi-supervised learning (McClosky et al., 2006; Spoustová et al., 2009). One of the main reasons for this inefficiency is attributed to the inefficiency of core classifiers trained with many feature combinations (*e.g.*, word n-grams). Hereafter, we refer to features that explicitly represent combinations of features as *conjunctive features* and the other atomic features as *primitive features*.

The feature combinations play an essential role in obtaining a classifier with state-of-the-art accuracy for several NLP tasks; recent examples include dependency parsing (Koo et al., 2008), parse re-ranking (McClosky et al., 2006), pronoun resolution (Nguyen and Kim, 2008), and semantic role labeling (Liu and Sarkar, 2007). However, 'explicit' feature combinations significantly increase the feature space, which slows down not only training but also testing of the classifier.

Kernel-based methods such as support vector machines (SVMs) consider feature combinations space-efficiently by using a polynomial kernel function (Cortes and Vapnik, 1995). The kernel-based classification is, however, known to be very slow in NLP tasks, so efficient classifiers should sum up the weights of the explicit conjunctive features (Isozaki and Kazawa, 2002; Kudo and Matsumoto, 2003; Goldberg and Elhadad, 2008).

ℓ_1-regularized log-linear models (ℓ_1-LLMs), on the other hand, provide sparse solutions, in which weights of irrelevant features are exactly zero, by assuming a Laplacian prior on the weights (Tibshirani, 1996; Kazama and Tsujii, 2003; Goodman, 2004; Gao et al., 2007). However, as Kazama and Tsujii (2005) have reported in a text categorization task and we later confirm in a dependency parsing task, when most features regarded as irrelevant during training ℓ_1-LLMs appear rarely in the task, we cannot greatly reduce the number of active features in each classification. In the end, when efficiency is a major concern, we must use exhaustive feature selection (Wu et al., 2007; Okanohara and Tsujii, 2009) or even restrict the order of conjunctive features at the expense of accuracy.

In this study, we provide a simple, but effective solution to the inefficiency of classifiers trained with higher-order conjunctive features (or polynomial kernel), by exploiting the Zipfian nature of language data. The key idea is to precompute the weights of primitive feature vectors and use them as partial results to compute the weight of a given feature vector. We use a trie called the *feature sequence trie* to efficiently find for a given feature vector its longest prefix feature vector whose weight has been computed. The trie is built from feature vectors generated by applying the classifier to actual data in the classification task. The time complexity of the classifier approaches time that

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1542–1551,
Singapore, 6-7 August 2009.

is linear with respect to the number of primitive features when the retrieved feature vector covers most of the features in the input feature vector.

We implemented our algorithm for SVM and LLM classifiers and evaluated the performance of the resulting classifiers in a Japanese dependency parsing task. Experimental results show that it successfully speeded up classifiers trained with higher-order conjunctive features by a factor of 10.

The rest of this paper is organized as follows. Section 2 introduces LLMs and SVMs. Section 3 proposes our classification algorithm. Section 4 presents experimental results. Section 5 concludes with a summary and addresses future directions.

2 Preliminaries

In this paper, we focus on linear classifiers that calculate the probability (or score) by summing up weights of individual features. Examples include not only log-linear models but also support vector machines with kernel expansion (Isozaki and Kazawa, 2002; Kudo and Matsumoto, 2003). Below, we introduce these two classifiers and their ways to consider feature combinations.

In classification-based NLP, the target task is modeled as one or more classification steps. For example in part-of-speech (POS) tagging, each classification decides whether to assign a particular *label* (POS tag) to a given *sample* (each word in a given sentence). Each sample is then represented by a *feature vector* $\boldsymbol{x}$, whose element x_i is a value of a feature function $f_i \in \mathcal{F}$.

Here, we assume a binary feature function $f_i(\boldsymbol{x}) \in \{0, 1\}$, in which a non-zero value means that particular context data appears in the sample. We say that a feature f_i is *active* in sample $\boldsymbol{x}$ when $x_i = f_i(\boldsymbol{x}) = 1$ and $|\boldsymbol{x}|$ represents the number of active features in $\boldsymbol{x}$ ($|\boldsymbol{x}| = |\{f_i | f_i(\boldsymbol{x}) = 1\}|$).

2.1 Log-Linear Models

The log-linear model (LLM), or also known as maximum-entropy model (Berger et al., 1996), is a linear classifier widely used in the NLP literature. Let the training data of LLMs be $\{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{L}$, where $\boldsymbol{x}_i \in \{0, 1\}^n$ is a feature vector and y_i is a class label associated with $\boldsymbol{x}_i$. We assume a binary label $y_i \in \{\pm 1\}$ here to simplify the argument.

The classifier provides conditional probability $p(y|\boldsymbol{x})$ for a given feature vector $\boldsymbol{x}$ and a label y:

$$p(y|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp \sum_i w_{i,y} f_{i,y}(\boldsymbol{x}, y), \quad (1)$$

where $f_{i,y}(\boldsymbol{x}, y)$ is a feature function that returns a non-zero value when $f_i(\boldsymbol{x}) = 1$ and the label is y, $w_{i,y} \in \mathbb{R}$ is a weight associated with $f_{i,y}$, and $Z(\boldsymbol{x}) = \sum_y \exp \sum_i w_{i,y} f_{i,y}(\boldsymbol{x}, y)$ is the partition function. We can consider feature combinations in LLMs by explicitly introducing a new conjunctive feature $f_{\mathcal{F}',y}(\boldsymbol{x}, y)$ that is activated when a particular set of features $\mathcal{F}' \subseteq \mathcal{F}$ to be combined is activated (namely, $f_{\mathcal{F}',y}(\boldsymbol{x}, y) = \bigwedge_{f_{i,y} \in \mathcal{F}'} f_{i,y}(\boldsymbol{x}, y)$).

We then introduce an ℓ_1-regularized LLM (ℓ_1-LLM), in which the weight vector $\boldsymbol{w}$ is tuned so as to maximize the logarithm of the a posteriori probability of the training data:

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{L} \log p(y_i|\boldsymbol{x}_i) - C\|\boldsymbol{w}\|_1. \quad (2)$$

Hyper-parameter C thereby controls the degree of over-fitting (solution sparseness). Interested readers may refer to the cited literature (Andrew and Gao, 2007) for the optimization procedures.

2.2 Support Vector Machines

A support vector machine (SVM) is a binary classifier (Cortes and Vapnik, 1995). Training with samples $\{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{L}$ where $\boldsymbol{x}_i \in \{0, 1\}^n$ and $y_i \in \{\pm 1\}$ yields the following decision function:

$$y(\boldsymbol{x}) = \mathrm{sgn}(g(\boldsymbol{x}) + b)$$
$$g(\boldsymbol{x}) = \sum_{\boldsymbol{x}_j \in \mathcal{SV}} y_j \alpha_j \phi(\boldsymbol{x}_j)^{\mathrm{T}} \phi(\boldsymbol{x}), \quad (3)$$

where $b \in \mathbb{R}$, $\phi : \mathbb{R}^n \mapsto \mathbb{R}^H$ and *support vectors* $\boldsymbol{x}_j \in \mathcal{SV}$ (subset of training samples), each of which is associated with weight $\alpha_j \in \mathbb{R}$. We hereafter call $g(\boldsymbol{x})$ the *weight function*. Nonlinear mapping function ϕ is chosen to make the training samples linearly separable in $\mathbb{R}^H$ space. Kernel function $k(\boldsymbol{x}_j, \boldsymbol{x}) = \phi(\boldsymbol{x}_j)^{\mathrm{T}} \phi(\boldsymbol{x})$ is then introduced to compute the dot product in $\mathbb{R}^H$ space without mapping $\boldsymbol{x}$ to $\phi(\boldsymbol{x})$.

To consider combinations of primitive features $f_j \in \mathcal{F}$, we use a *polynomial kernel* $k_d(\boldsymbol{x}_j, \boldsymbol{x}) = (\boldsymbol{x}_j^{\mathrm{T}} \boldsymbol{x} + 1)^d$. From Eq. 3, we obtain the weight function for the polynomial kernel as:

$$g(\boldsymbol{x}) = \sum_{\boldsymbol{x}_j \in \mathcal{SV}} y_j \alpha_j (\boldsymbol{x}_j^{\mathrm{T}} \boldsymbol{x} + 1)^d. \quad (4)$$

Since we assumed that x_i is a binary value representing whether a (primitive) feature f_i is active in the sample, the polynomial kernel of degree d implies a mapping ϕ_d from $\boldsymbol{x}$ to $\phi_d(\boldsymbol{x})$ that has

$H = \sum_{k=0}^{d} \binom{n}{k}$ dimensions. Each dimension represents a (weighted) conjunction of d features in the original sample $\boldsymbol{x}$.[1]

Kernel Expansion (SVM-KE) The time complexity of Eq. 4 is $O(|\boldsymbol{x}| \cdot |\mathcal{SV}|)$. This cost is usually high for classifiers used in NLP tasks because they often have many support vectors ($|\mathcal{SV}| > 10,000$). *Kernel expansion* (KE) was proposed by Isozaki and Kazawa (2002) to convert Eq. 4 into the linear sum of the weights in the mapped feature space as in LLM ($p(y|\boldsymbol{x})$ in Eq. 1):

$$g(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}^d = \sum_i w_i x_i^d, \tag{5}$$

where $\boldsymbol{x}^d$ is a binary feature vector whose element x_i^d has a non-zero value when $(\phi_d(\boldsymbol{x}))_i > 0$, $\boldsymbol{w}$ is the weight vector for $\boldsymbol{x}^d$ in the expanded feature space $\mathcal{F}^d$ and is precalculated from the support vectors $\boldsymbol{x}_j$ and their weights α_j. Interested readers may refer to Kudo and Matsumoto (2003) for the detailed computation for obtaining $\boldsymbol{w}$.

The time complexity of Eq. 5 (and Eq. 1) is $O(|\boldsymbol{x}^d|)$, which is linear with respect to the number of active features in $\boldsymbol{x}^d$ within the expanded feature space $\mathcal{F}^d$.

Heuristic Kernel Expansion (SVM-HKE) To make the weight vector sparse, Kudo and Matsumoto (2003) proposed a heuristic method that filters out less useful features whose absolute weight values are less than a pre-defined threshold σ.[2] They reported that increased threshold value σ resulted in a dramatically sparse feature space $\mathcal{F}^d$, which had the side-effects of accuracy degradation and classifier speed-up.

3 Proposed Method

In this section, we propose a method that speeds up a classifier trained with many conjunctive features. Below, we focus on a kernel-based classifier trained with a polynomial kernel of degree d (here, SVMs), but an analogous argument is possible for linear classifiers (*e.g.*, LLMs).[3]

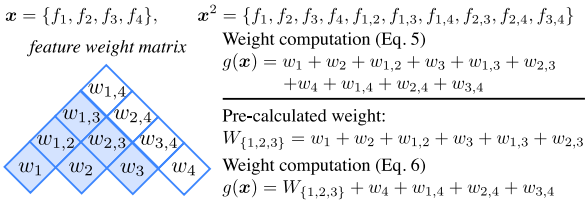


Figure 1: Efficient computation of $g(\boldsymbol{x})$.

We hereafter represent a binary feature vector $\boldsymbol{x}$ as a set of active features $\{f_i | f_i(\boldsymbol{x}) = 1\}$. $\boldsymbol{x}$ can thereby be represented as an element of the power set $2^{\mathcal{F}}$ of the set of features $\mathcal{F}$.

3.1 Idea

Let us remember that weight function $g(\boldsymbol{x})$ in Eq. 5 maps $\boldsymbol{x} \in 2^{\mathcal{F}}$ to $W \in \mathbb{R}$. If we could calculate $W_{\boldsymbol{x}} = g(\boldsymbol{x})$ for all possible $\boldsymbol{x}$ in advance, we could obtain $g(\boldsymbol{x})$ by simply checking $|\boldsymbol{x}|$ elements, namely, in $O(|\boldsymbol{x}|)$ time. However, because $|\{\boldsymbol{x} | \boldsymbol{x} \in 2^{\mathcal{F}}\}| = 2^{|\mathcal{F}|}$ and $|\mathcal{F}|$ is likely to be very large (often $|\mathcal{F}| > 10,000$ in NLP tasks), this calculation is impractical.

We then compute and store weight $W_{\boldsymbol{x}'} = g(\boldsymbol{x}')$ for $\boldsymbol{x}' \in \mathcal{V}_c (\subset 2^{\mathcal{F}})$, a certain subset of the possible value space, and compute $g(\boldsymbol{x})$ for $\boldsymbol{x} \notin \mathcal{V}_c$ by using precalculated weight $W_{\boldsymbol{x}_c}$ for $\boldsymbol{x}_c \subseteq$[4] $\boldsymbol{x}$ in the following way:

$$g(\boldsymbol{x}) = W_{\boldsymbol{x}_c} + \sum_{f_i \in \boldsymbol{x}^d - \boldsymbol{x}_c^d} w_i. \tag{6}$$

Intuitively speaking, starting from partial weight $W_{\boldsymbol{x}_c}$, we add up remaining weights of primitive features $f \in \mathcal{F}$ that are not active in $\boldsymbol{x}_c$ but active in $\boldsymbol{x}$ and conjunctive features that combine f and the other active features in $\boldsymbol{x}$.

An example of this computation ($d = 2$) is depicted in Figure 1. We can efficiently compute $g(\boldsymbol{x})$ for a vector $\boldsymbol{x}$ that has four active features f_1, f_2, f_3, and f_4 (and $\boldsymbol{x}^2$ has their six conjunctive features) using precalculated weight $W_{\{1,2,3\}}$; we should first check the three features f_1, f_2, and f_3 to retrieve $W_{\{1,2,3\}}$ and next check the remaining four features related to f_4, namely f_4, $f_{1,4}$, $f_{2,4}$, and $f_{3,4}$, in order to add up the remaining

[1] For example, given an input vector $\boldsymbol{x} = (x_1, x_2)^{\mathrm{T}}$ and a support vector $\boldsymbol{x}' = (x_1', x_2')^{\mathrm{T}}$, the 2nd-order polynomial kernel returns $k_2(\boldsymbol{x}', \boldsymbol{x}) = (x_1' x_1 + x_2' x_2 + 1)^2 = 3x_1' x_1 + 3x_2' x_2 + 2x_1' x_1 x_2' x_2 + 1$ ($\because x_i', x_i \in \{0, 1\}$). This function thus implies a mapping $\phi_2(\boldsymbol{x}) = (1, \sqrt{3}x_1, \sqrt{3}x_2, \sqrt{2}x_1 x_2)^{\mathrm{T}}$. In the following argument, we ignore the dimension of the constant in the mapped space and assume constant b is set to include it.

[2] Precisely speaking, they set different thresholds to positive ($\alpha_j > 0$) and negative ($\alpha_j < 0$) support vectors, considering the proportion of positive and negative support vectors.

[3] When a feature vector $\boldsymbol{x}$ includes (explicit) conjunctive features $f \in \mathcal{F}^d$, we assume weight function $g'(y|\boldsymbol{x}') = g(y|\boldsymbol{x})$, where $\boldsymbol{x}'$ is a projection of $\boldsymbol{x}$ (by $\phi_d^{-1} : \mathcal{F}^d \to \mathcal{F}$).

[4] This means that all active features in $\boldsymbol{x}_c$ are active in $\boldsymbol{x}$.

weights, while the normal computation in Eq. 5 should check the four primitive and six conjunctive features to get the individual weights.

Expected time complexity Counting the number of features to be checked in the computation, we obtain the time complexity $f(\boldsymbol{x}, d)$ of Eq. 6 as:

$$f(\boldsymbol{x}, d) = O(|\boldsymbol{x}_c| + |\boldsymbol{x}^d| - |\boldsymbol{x}_c^d|), \quad (7)$$

$$\textit{where} \quad |\boldsymbol{x}^d| = \sum_{k=1}^{d} \binom{|\boldsymbol{x}|}{k} \quad (8)$$

(*e.g.*, $|\boldsymbol{x}^2| = \frac{|\boldsymbol{x}|^2+|\boldsymbol{x}|}{2}$ and $|\boldsymbol{x}^3| = \frac{|\boldsymbol{x}|^3+5|\boldsymbol{x}|}{6}$).[5] Note that when $|\boldsymbol{x}_c|$ becomes close to $|\boldsymbol{x}|$, this time complexity actually approaches $O(|\boldsymbol{x}|)$.

Thus, to minimize this computational cost, $\boldsymbol{x}_c$ is to be chosen from $\mathcal{V}_c$ as follows:

$$\boldsymbol{x}_c = \underset{\boldsymbol{x}' \in \mathcal{V}_c, \boldsymbol{x}' \subseteq \boldsymbol{x}}{\operatorname{argmin}} (|\boldsymbol{x}'| + |\boldsymbol{x}^d| - |\boldsymbol{x}'^d|). \quad (9)$$

3.2 Construction of Feature Sequence Trie

There are two issues with speeding up the classifier by the computation shown in Eq. 6. First, since we can store weights for only a small fraction of possible feature vectors (namely, $|\mathcal{V}_c| \ll 2^{|\mathcal{F}|}$), we should choose $\mathcal{V}_c$ so as to maximize its impact on the speed-up. Second, we should quickly find an optimal $\boldsymbol{x}_c$ from $\mathcal{V}_c$ for a given feature vector $\boldsymbol{x}$.

The solution to the first problem is to enumerate partial feature vectors that frequently appear in the target task. Note that typical linguistic features used in NLP tasks usually consist of disjunctive sets of features (*e.g.*, word surface and POS), in which each set is likely to follow Zipf's law (Zipf, 1949) and correlate with each other. We can expect the distribution of feature vectors, the mixture of Zipf distributions, to be Zipfian. This has been confirmed for word n-grams (Egghe, 2000) and itemset support distribution (Chuang et al., 2008). We can thereby expect that a small set of partial feature vectors commonly appear in the task.

To solve the second problem, we introduce a *feature sequence trie* (*fstrie*), which represents a hierarchy of feature vectors, to enable the classifier to efficiently retrieve (sub-)optimal $\boldsymbol{x}_c$ (in Eq. 9) for a given feature vector $\boldsymbol{x}$. We build an fstrie in the following steps:

Step 1: Apply the target classifier to actual (raw) data in the task to enumerate possible feature vectors (hereafter, *source feature vectors*).

[5] This is the maximum number of conjunctive features.

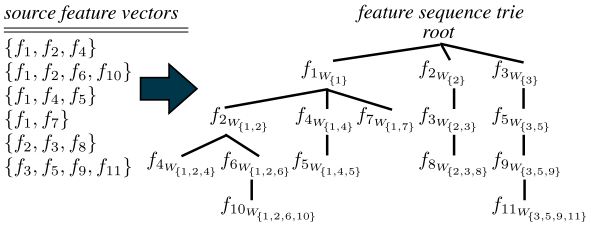


Figure 2: Feature sequence trie and completion of prefix feature vector weights.

Step 2: Sort the features in each source feature vector according to their frequency in the training data (in descending order).

Step 3: Build a trie from the source feature vectors by regarding feature indices as characters and store weights of all prefix feature vectors.

An fstrie built from six source feature vectors is shown in Figure 2. In fstries, a path from the root to another node represents a feature vector. An important point here is that the fstrie stores the weights of all prefix feature vectors of the source feature vectors, and the trie structure enables us to retrieve for a given feature vector $\boldsymbol{x}$ the weight of its longest prefix vector $\boldsymbol{x}_c \subseteq \boldsymbol{x}$ in $O(|\boldsymbol{x}_c|)$ time. To handle feature functions in LLMs (Eq. 1), we store partial weight $W_{\boldsymbol{x}_c,y} = \sum_i w_{i,y} f_{i,y}(\boldsymbol{x}_c, y)$ for each label y on the node that expresses $\boldsymbol{x}_c$.

Since we sort the features in the source feature vectors according to their frequency, the prefix feature vectors exclude less frequent features in the source feature vectors. Lexical features or finer-grained features (*e.g.*, POS-subcategory) are usually less frequent than coarse-grained features (*e.g.*, POS), so they lie in the latter part of the feature vectors. This sorting helps us to retrieve longer feature vector $\boldsymbol{x}_c$ for input feature vector $\boldsymbol{x}$ that will have diverse infrequent features. It also minimizes the size of fstrie by sharing the common frequent prefix (*e.g.*, $\{f_1, f_2\}$ in Figure 2).

Pruning nodes from fstrie We have so far described the way to construct an fstrie from the source feature vectors. However, a naive enumeration of source feature vectors will result in the explosion of the fstrie size, and we want to have a principled way to control the fstrie size rather than reducing the processed data size. Below, we present a method that prunes useless prefix feature vectors (nodes) from the constructed fstrie to maximize its impact on the classifier efficiency.

Algorithm 1 PRUNE NODES FROM FSTRIE

Input: fstrie T, node_limit $N \in \mathbb{N}$
Output: fstrie T

1: **while** # of nodes in $T > N$ **do**
2: $\quad \boldsymbol{x}_c \leftarrow \underset{\boldsymbol{x}' \in \mathit{leaf}(T)}{\operatorname{argmin}}\ u(\boldsymbol{x}')$
3: $\quad$ remove $\boldsymbol{x}_c$, T
4: **end while**
5: **return** T

We adopt a greedy strategy that iteratively prunes a leaf node (one prefix feature vector and its weight) from the fstrie built from all the source feature vectors, according to a certain utility score calculated for each node. In this study, we consider two metrics for each prefix feature vector $\boldsymbol{x}_c$ to calculate its utility score.

Probability $p(\boldsymbol{x}_c)$**,** which denotes how often the stored weight $W_{\boldsymbol{x}_c}$ will be used in the target task. The maximum-likelihood estimation provides probability:

$$p(\boldsymbol{x}_c) = \frac{\sum_{\boldsymbol{x}' \supseteq \boldsymbol{x}_c} n_{\boldsymbol{x}'}}{\sum_{\boldsymbol{x}} n_{\boldsymbol{x}}}, \qquad (10)$$

where $n_{\boldsymbol{x}} \in \mathbb{N}$ is the frequency count of a source feature vector $\boldsymbol{x}$ in the processed data.

Computation reduction $\Delta_d(\boldsymbol{x}_c)$**,** which denotes how much computation is reduced by $W_{\boldsymbol{x}_c}$ to calculate a weight of $\boldsymbol{x} \supseteq \boldsymbol{x}_c$. This can be estimated by counting the number of conjunctive features we additionally have to check when we remove $\boldsymbol{x}_c$. Since the fstrie stores the weight of a prefix feature vector $\boldsymbol{x}_{c\text{-}} \subset \boldsymbol{x}_c$ such that $|\boldsymbol{x}_{c\text{-}}| = |\boldsymbol{x}_c| - 1$ (*e.g.*, in Figure 2, $\boldsymbol{x}_{c\text{-}} = \{f_1, f_2\}$ for $\boldsymbol{x}_c = \{f_1, f_2, f_4\}$), we can define the computation reduction as:

$$\begin{aligned}\Delta_d(\boldsymbol{x}_c) &= (|\boldsymbol{x}_c^d| - |\boldsymbol{x}_{c\text{-}}^d|) - (|\boldsymbol{x}_c| - |\boldsymbol{x}_{c\text{-}}|) \\ &= \sum_{k=2}^{d} \binom{|\boldsymbol{x}_c|}{k} - \sum_{k=2}^{d} \binom{|\boldsymbol{x}_c| - 1}{k} \\ &\quad (\because \text{Eq. 8}).\end{aligned}$$

$$\Delta_2(\boldsymbol{x}_c) = |\boldsymbol{x}_c| - 1 \text{ and } \Delta_3(\boldsymbol{x}_c) = \tfrac{|\boldsymbol{x}_c|^2 - |\boldsymbol{x}_c|}{2}.$$

We calculate utility score of each node $\boldsymbol{x}_c$ in the fstrie as $u(\boldsymbol{x}_c) = p(\boldsymbol{x}_c) \cdot \Delta_d(\boldsymbol{x}_c)$, which means the expected computation reduction by $\boldsymbol{x}_c$ in the target task, and prune the lowest-utility-score leaf nodes from the fstrie one by one (Algorithm 1). If several prefix vectors have the same utility score, we eliminate them in numerical descending order.

Algorithm 2 COMPUTE WEIGHT WITH FSTRIE

Input: fstrie T, weight vector $\boldsymbol{w} \in \mathbb{R}^{|\mathcal{F}^d|}$
feature vector $\boldsymbol{x} \in 2^{\mathcal{F}}$
Output: weight $W = g(\boldsymbol{x}) \in \mathbb{R}$

1: $\boldsymbol{x} \leftarrow \text{sort}(\boldsymbol{x})$
2: $\langle \boldsymbol{x}_c, W_{\boldsymbol{x}_c} \rangle \leftarrow$ prefix_search(T, $\boldsymbol{x}$)
3: $W \leftarrow W_{\boldsymbol{x}_c}$
4: **for all** feature $f_j \in \boldsymbol{x}^d - \boldsymbol{x}_c^d$ **do**
5: $\quad W \leftarrow W + w_j$
6: **end for**
7: **return** W

3.3 Classification Algorithm

Our classification algorithm is shown in detail in Algorithm 2. The classifier first sorts the active features in input feature vector $\boldsymbol{x}$ according to their frequency in the training data. Then, for $\boldsymbol{x}$, it retrieves the longest common prefix vector $\boldsymbol{x}_c$ from the fstrie (line 2 in Algorithm 2). It then adds the weights of the remaining features to partial weight $W_{\boldsymbol{x}_c}$ (line 5 in Algorithm 2).

Note that the remaining features whose weights we sum up (line 4 in Algorithm 2) are primitive and conjunctive features that relate to $f \in \boldsymbol{x} - \boldsymbol{x}_c$, which appear less frequently than $f' \in \boldsymbol{x}_c$ in the training data. Thus, when we apply our algorithm to classifiers with the sparse solution (*e.g.*, SVM-HKEs or ℓ_1-LLMs), $|\boldsymbol{x}^d| - |\boldsymbol{x}_c^d|$ can be much smaller than the theoretical expectation (Eq. 8). We confirmed this in the following experiments.

4 Evaluation

We applied our algorithm to SVM-KE, SVM-HKE, and ℓ_1-LLM classifiers and evaluated the resulting classifiers in a Japanese dependency parsing task. To the best of our knowledge, there are no previous reports of an *exact* weight calculation faster than linear summation (Eqs. 1 and 5). We also compared our SVM classifier with a classifier called polynomial kernel inverted (PKI: Kudo and Matsumoto (2003)), which uses the polynomial kernel (Eq. 4) and inverted indexing to support vectors.

4.1 Experimental Settings

A Japanese dependency parser inputs *bunsetsu*-segmented sentences and outputs the correct head (bunsetsu) for each bunsetsu; here, a bunsetsu is a grammatical unit in Japanese consisting of one or more content words followed by zero or more function words. A parser generates a feature vec-

Modifier, modifiee bunsetsu	head word (surface-form, POS, POS-subcategory, inflection form), functional word (surface-form, POS, POS-subcategory, inflection form), brackets, quotation marks, punctuation marks, position in sentence (beginning, end)
Between bunsetsus	distance (1, 2–5, 6–), case-particles, brackets, quotation marks, punctuation marks

Table 1: Feature set used for experiments.

tor for a particular pair of bunsetsus (modifier and modifiee candidates) by exploiting the head-final and projective (Nivre, 2003) nature of dependency relations in Japanese. The classifier then outputs label y = '$+1$' (dependent) or '-1' (independent).

Since our classifier is independent of individual parsing algorithms, we targeted speeding up (a classifier in) the shift-reduce parser proposed by Sassano (2004), which has been reported to be the most efficient for this task, with almost state-of-the-art accuracy (Iwatate et al., 2008). This parser decreases the number of classification steps by using the fact that a bunsetsu is likely to modify a bunsetsu close to itself. Due to space limitations, we omit the details of the parsing algorithm.

We used the standard feature set tailored for this task (Kudo and Matsumoto, 2002; Sassano, 2004; Iwatate et al., 2008) (Table 1). Note that features listed in the 'Between bunsetsus' row represent contexts between the target pair of bunsetsus and appear independently from other features, which will become an obstacle to finding the longest prefix vector. This task is therefore a better measure of our method than simple sequential labeling such as POS tagging or named-entity recognition.

For evaluation, we used Kyoto Text Corpus Version 4.0 (Kurohashi and Nagao, 2003), Mainichi news articles in 1995 that have been manually annotated with dependency relations.[6] The training, development, and test sets included 24,283, 4833, and 9284 sentences, and 234,685, 47,571, and 89,874 bunsetsus, respectively. The training samples generated from the training set included 150,064 positive and 146,712 negative samples.

The following experiments were performed on a server with an Intel® Xeon™ 3.20-GHz CPU. We used TinySVM[7] and a simple C++ library for maximum entropy classification[8] to train SVMs and ℓ_1-LLMs, respectively. We used Darts-Clone,[9]

[6] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/corpus-e.html
[7] http://chasen.org/~taku/software/TinySVM/
[8] http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/maxent/
[9] http://code.google.com/p/darts-clone/

Model type			Model statistics		Dep.	Sent.
Model	d	ω / σ	$\lvert\mathcal{F}^d\rvert$	$\lvert\boldsymbol{x}^d\rvert$	acc.	acc.
SVM-KE	1	0	39712	27.3	88.29	46.49
SVM-KE	2	0	1478109	380.6	90.76	53.83
SVM-KE	3	0	26194354	3286.7	90.93$^{\gg}$	54.43$^{\gg}$
SVM-HKE	3	0.001	13247675	2725.9	90.92$^{\gg}$	54.39$^{\gg}$
SVM-HKE	3	0.002	2514385	2238.1	90.91$^{\gg}$	54.32$^{>}$
SVM-HKE	3	0.003	793195	1855.4	90.83	54.21
SVM-KE	4	0	293416102	20395.4	90.91$^{\gg}$	**54.69**$^{\gg}$
SVM-HKE	4	0.0002	96522236	15282.1	90.93$^{\gg}$	54.53$^{>}$
SVM-HKE	4	0.0004	19245076	11565.0	**90.96**$^{\gg}$	54.64$^{\gg}$
SVM-HKE	4	0.0006	7277592	8958.2	90.84	54.48$^{>}$
ℓ_1-LLM	1	1.0	9268	26.5	88.22	46.06
ℓ_1-LLM	2	2.0	32575	309.8	90.62	53.46
ℓ_1-LLM	3	3.0	129503	2088.3	90.71	54.09$^{>}$
ℓ_1-LLM	3	4.0	85419	1803.0	90.61	53.79
ℓ_1-LLM	3	5.0	63046	1699.5	90.59	53.55

Table 2: Specifications of LLMs and SVMs. The accuracy marked with '$\gg$' or '$>$' was significantly better than the $d = 2$ counterpart ($p < 0.01$ or $0.01 \leq p < 0.05$ by McNemar's test).

a double-array trie (Aoe, 1989; Yata et al., 2008), as a compact trie implementation. All these libraries and algorithms are implemented in C++. The code for building fstries occupies 100 lines, while the code for the classifier occupies 20 lines (except those for kernel expansion).

4.2 Results

Specifications of SVMs and LLMs used here are shown in Table 2; $|\mathcal{F}^d|$ is the number of active features, while $|\boldsymbol{x}^d|$ is the average number of active features in each classification for the test corpus. Dependency accuracy is the ratio of dependency relations correctly identified by the parser, while sentence accuracy is the exact match accuracy of complete dependency relations in a sentence.

For LLM training, we designed explicit conjunctive features for all the d or lower-order feature combinations to make the results comparable to those of SVMs. We could not train $d = 4$ LLMs due to parameter explosion. We varied SVM soft margin parameter c from 0.1 to 0.000001 and LLM width factor parameter ω,[10] which controls the impact of the prior, from 1.0 to 5.0, and adjusted the values to maximize dependency accuracy for the development set: $(d, c) = (1, 0.1), (2, 0.005), (3, 0.0001), (4, 0.000005)$ for SVMs and $(d, \omega) = (1, 1.0), (2, 2.0), (3, 4.0)$ for ℓ_1-LLMs.

The accuracy of around 90.9% (SVM-KE, $d = 3, 4$) is close to the performance of state-of-the-

[10] The parameter C of ℓ_1-LLM in Eq. 2 was set to ω/L (referred to in Kazama and Tsujii (2003) as '*single width*').

Model type	d	PKI classify [ms/sent.]	Baseline Mem. (MB)	Baseline Time [ms/sent.] classify (total)	Proposed w/ fstrie$_S$ Mem. (MB)	Proposed w/ fstrie$_S$ Time [ms/sent.] classify (total)	Proposed w/ fstrie$_M$ Mem. (MB)	Proposed w/ fstrie$_M$ Time [ms/sent.] classify (total)	Proposed w/ fstrie$_L$ Mem. (MB)	Proposed w/ fstrie$_L$ Time [ms/sent.] classify (total)	Speed up
SVM-KE	1	13.480	0.2	**0.003** (0.015)	+0.6	0.006 (0.018)	+20.2	0.007 (0.018)	+662.9	0.016 (0.029)	NA
SVM-KE	2	10.313	13.5	0.041 (0.054)	+0.5	**0.020** (0.032)	+18.0	0.021 (0.034)	+662.4	0.023 (0.036)	2.1
SVM-KE	3	10.945	142.2	0.345 (0.361)	+0.5	0.163 (0.178)	+18.2	0.108 (0.123)	+667.0	**0.079** (0.093)	4.4
SVM-KE	4	12.603	648.0	2.338 (2.363)	+0.5	1.156 (1.178)	+18.6	0.671 (0.690)	+675.9	**0.415** (0.432)	5.6

Table 3: Parsing results for test corpus: SVM-KE classifiers with dense feature space.

art parsers (Iwatate et al., 2008), and the model statistics are considered to be complex (or realistic) enough to evaluate our classifier's utility. The number of support vectors of SVMs was $71,766 \pm 9.2\%$, which is twice as many as those used by Kudo and Matsumoto (2003) (34,996) in their experiments on the same task.

We could clearly observe that the number of active features $|\boldsymbol{x}^d|$ increased dramatically according to the order d of feature combinations. The density of $|\boldsymbol{x}^d|$ for SVMs was very high (*e.g.*, $|\boldsymbol{x}^3| = 3286.7$, close to the maximum shown in Eq. 8: $(27.3^3 + 5 \times 27.3)/6 \simeq 3414$.

For $d \geq 3$ models, we attempted to control the size of the feature space $|\mathcal{F}^d|$ by changing the model's hyper-parameters: threshold σ for the SVM-HKE and width factor ω for the ℓ_1-LLM. Although we successfully reduced the size of the feature space $|\mathcal{F}^d|$, we could not dramatically reduce the average number of active features $|\boldsymbol{x}^d|$ in each classification while keeping the accuracy advantage. This confirms that the solution sparseness does not suffice to obtain an efficient classifier.

We obtained source feature vectors to build fstries by applying parsers with the target classifiers to a raw corpus in the target domain, 3,258,313 sentences of 1991–94 Mainichi news articles that were morphologically analyzed by JUMAN[6] and segmented into bunsetsus by KNP.[6] We first built fstrie$_L$ using all the source feature vectors. We then attempted to reduce the number of prefix feature vectors in fstrie$_L$ to $1/2^n$ the size by Algorithm 1. We refer to fstries built from 1/32 and 1/1024 of the prefix feature vectors in fstrie$_L$ as fstrie$_M$ and fstrie$_S$ in the following experiments.

Because we exploited Algorithm 2 to calculate the weights of the prefix feature vectors, it took less than one hour (59 min. 29 sec.) on the 3.20-GHz server to build fstrie$_L$ (and calculate the utility score for all the nodes in it) for the slowest SVM-KE ($d = 4$) from the 40,409,190 source feature vectors (62,654,549 prefix feature vectors) generated by parsing the 3,258,313 sentences.

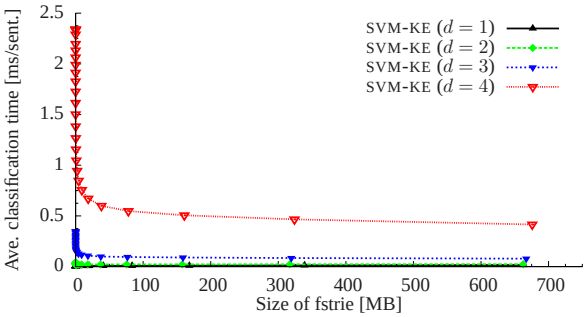


Figure 3: Average classification time per sentence plotted against size of fstrie: SVM-KE.

Results for SVM-KE with dense feature space The performances of parsers having SVM-KE classifiers with and without the fstrie are given in Table 3. The 'speed-up' column shows the speed-up factor of the most efficient classifier (bold) versus the baseline classifier without fstries. Since each classifier solved a slightly different number of classification steps ($112,853 \pm 0.15\%$), we show the (average) cumulative classification time for a sentence. The Mem. columns show the size of weight vectors for SVM-KE classifiers and the size of fstries$_S$, fstries$_M$, and fstries$_L$, respectively.

The fstries successfully speeded up SVM-KE classifiers with the dense feature space.[11] The SVM-KE classifiers without fstries were still faster than PKI, but as expected from a large $|\boldsymbol{x}^d|$ value, the classifiers with higher conjunctive features were much slower than the classifier with only primitive features by factors of 13 ($d = 2$), 109 ($d = 3$) and 738 ($d = 4$) and the classification time accounted for most of the parsing time.

The average classification time of our classifiers plotted against fstrie size is shown in Figure 3. Surprisingly, we obtained a significant speed-up even with tiny fstrie sizes of < 1 MB. Furthermore, we naively controlled the fstrie size by sim-

[11] The inefficiency of the classifier ($d = 1$) results from the cost of the additional sort function (line 1 in Algorithm 2) and CPU cache failure due to random accesses to the huge fstries.

Model type	d	σ / ω	Baseline Mem. (MB)	Baseline Time [ms/sent.] classify (total)	Proposed w/ fstrie$_S$ Mem. (MB)	Proposed w/ fstrie$_S$ Time [ms/sent.] classify (total)	Proposed w/ fstrie$_M$ Mem. (MB)	Proposed w/ fstrie$_M$ Time [ms/sent.] classify (total)	Proposed w/ fstrie$_L$ Mem. (MB)	Proposed w/ fstrie$_L$ Time [ms/sent.] classify (total)	Speed up
SVM-HKE	3	0.001	64.6	0.348 (0.363)	+0.5	0.151 (0.166)	+17.6	0.097 (0.111)	+638.0	**0.070** (0.084)	5.0
SVM-HKE	3	0.002	13.9	0.332 (0.346)	+0.5	0.123 (0.137)	+17.0	0.074 (0.088)	+612.2	**0.053** (0.067)	6.2
SVM-HKE	3	0.003	4.2	0.314 (0.328)	+0.4	0.102 (0.115)	+14.7	0.057 (0.070)	+526.2	**0.041** (0.054)	7.8
SVM-HKE	4	0.0002	235.0	2.258 (2.280)	+0.5	1.022 (1.042)	+17.7	0.558 (0.575)	+637.1	**0.330** (0.346)	6.8
SVM-HKE	4	0.0004	82.8	2.038 (2.058)	+0.5	0.816 (0.835)	+16.8	0.414 (0.430)	+601.7	**0.234** (0.249)	8.7
SVM-HKE	4	0.0006	32.2	1.802 (1.820)	+0.4	0.646 (0.662)	+15.7	0.311 (0.326)	+558.9	**0.168** (0.183)	10.7
ℓ_1-LLM	1	1.0	0.1	**0.004** (0.016)	+0.8	0.006 (0.018)	+25.0	0.007 (0.019)	+787.7	0.016 (0.029)	NA
ℓ_1-LLM	2	2.0	0.4	0.043 (0.055)	+0.6	0.016 (0.028)	+20.5	**0.015** (0.027)	+698.0	0.018 (0.030)	2.9
ℓ_1-LLM	3	3.0	1.0	0.314 (0.326)	+0.5	0.091 (0.103)	+17.8	0.041 (0.054)	+601.0	**0.027** (0.040)	11.6
ℓ_1-LLM	3	4.0	0.7	0.300 (0.313)	+0.5	0.082 (0.094)	+16.3	0.036 (0.049)	+550.1	**0.024** (0.037)	12.4
ℓ_1-LLM	3	5.0	0.5	0.290 (0.302)	+0.5	0.076 (0.088)	+15.1	0.032 (0.045)	+510.7	**0.022** (0.035)	13.3

Table 4: Parsing results for test corpus: SVM-HKE and ℓ_1-LLM classifiers with sparse feature space.

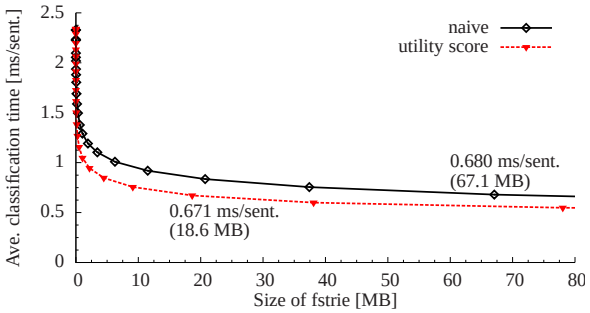


Figure 4: Fstrie reduction: utility score vs. processed sentence reduction for SVM-KE ($d = 4$).

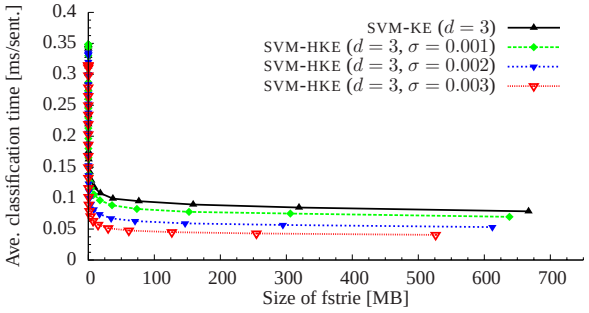


Figure 5: Average classification time per sentence plotted against size of fstrie: SVM-HKE ($d = 3$).

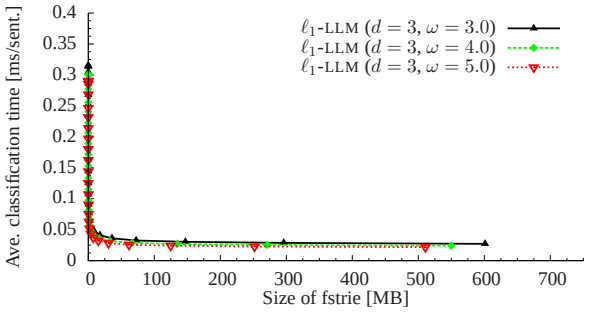


Figure 6: Average classification time per sentence plotted against size of fstrie: ℓ_1-LLM ($d = 3$).

ply reducing the number of sentences processed to $1/2^n$. The impact on the speed-up of the resulting fstries (*naive*) and the fstries constructed by our utility score (*utility-score*) on SVM-KE ($d = 4$) is shown in Figure 4. The Zipfian nature of language data let us obtain a substantial speed-up even when we naively reduced the fstrie size, and the utility score further decreased the fstrie size required to obtain the same speed-up. We needed less than 1/3 size fstries to achieve the same speed-up: 0.671 ms./sent. (18.6 MB) (*utility-score*) vs. 0.680 ms./sent. (67.1 MB) (*naive*).

Results for SVM-HKE and ℓ_1-LLM classifiers with sparse feature space The performances of parsers having SVM-HKE and ℓ_1-LLM classifiers with and without the fstrie are given in Table 4. The fstries successfully speeded up the SVM-HKE and ℓ_1-LLM classifiers by factors of 10.7 (SVM-HKE, $d = 4$, $\sigma = 0.0006$) and 11.6 (ℓ_1-LLM, $d = 3$, $\omega = 3.0$). We obtained more speed-up when we used fstries for classifiers with more sparse feature space $\mathcal{F}^d$ (Figures 5 and 6). The parsing speed with $d = 3$ models are now comparable to the parsing speed with $d = 2$ models.

Without fstries, little speed-up of SVM-HKE classifiers versus the SVM-KE classifiers (in Table 3) was obtained due to the mild reduction in the average number of active features $|\boldsymbol{x}^d|$ in the classification. This result conforms to the results reported in (Kudo and Matsumoto, 2003).

The parsing speed reached 14,937 sentences per second with accuracy of 90.91% (SVM-HKE, $d = 3$, $\sigma = 0.002$). We used this parser to process 1,005,918 sentences (5,934,184 bunsetsus) randomly extracted from Japanese weblog feeds

updated in November 2008, to see how much the impact of fstries lessens when the test data and the data processed to build fstries mismatch. The parsing time was 156.4 sec. without fstrie$_{\mathrm{L}}$, while it was just 35.9 sec. with fstrie$_{\mathrm{L}}$. The speed-up factor of 4.4 on weblog feeds was slightly worse than that on news articles ($0.346/0.067 = 5.2$) but still evident. This implies that sorting features in building fstries yielded prefix features vectors that commonly appear in this task, by excluding domain-specific features such as lexical features.

In summary, our algorithm successfully minimized the efficiency gap among classifiers with different degrees of feature combinations and made accurate classifiers trained with higher-order feature combinations practical.

5 Conclusion and Future Work

Our simple method speeds up a classifier trained with many conjunctive features by using precalculated weights of (partial) feature vectors stored in a feature sequence trie (fstrie). We experimentally demonstrated that it speeded up SVM and LLM classifiers for a Japanese dependency parsing task by a factor of 10. We also confirmed that the sparse feature space provided by ℓ_1-LLMs and SVM-HKEs contributed much to size reduction of the fstrie required to achieve the same speed-up. The implementations of the proposed algorithm for LLMs and SVMs (with a polynomial kernel) and the Japanese dependency parser will be available at http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/.

We plan to apply our method to wider range of classifiers used in various NLP tasks. To speed up classifiers used in a real-time application, we can build fstries incrementally by using feature vectors generated from user inputs. When we run our classifiers on resource-tight environments such as cell-phones, we can use a random feature mixing technique (Ganchev and Dredze, 2008) or a memory-efficient trie implementation based on a succinct data structure (Jacobson, 1989; Delpratt et al., 2006) to reduce required memory usage.

We will combine our method with other techniques that provide sparse solutions, for example, kernel methods on a budget (Dekel and Singer, 2007; Dekel et al., 2008; Orabona et al., 2008) or kernel approximation (surveyed in Kashima et al. (2009)). It is also easy to combine our method with SVMs with partial kernel expansion (Goldberg and Elhadad, 2008), which will yield slower but more space-efficient classifiers. We will in the future consider an issue of speeding up decoding with structured models (Lafferty et al., 2001; Miyao and Tsujii, 2002; Sutton et al., 2004).

Acknowledgment The authors wish to thank Susumu Yata and Yoshimasa Tsuruoka for letting the authors to use their pre-release libraries. The authors also thank Nobuhiro Kaji and the anonymous reviewers for their valuable comments.

References

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L_1-regularized log-linear models. In *Proc. ICML 2007*, pages 33–40.

Jun'ichi Aoe. 1989. An efficient digital search algorithm by using a double-array structure. *IEEE Transactions on Software Engineering*, 15(9):1066–1077, September.

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

Kun-Ta Chuang, Jiun-Long Huang, and Ming-Syan Chen. 2008. Power-law relationship and self-similarity in the itemset support distribution: analysis and applications. *The VLDB Journal*, 17(5):1121–1141, August.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297, September.

Ofer Dekel and Yoram Singer. 2007. Support vector machines on a budget. In Bernhard Schölkopf, John Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems 19*, pages 345–352. The MIT Press.

Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. 2008. The forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, 37(5):1342–1372, January.

O'Neil Delpratt, Naila Rahman, and Rajeev Raman. 2006. Engineering the LOUDS succinct tree representation. In *Proc. WEA 2006*, pages 134–145.

Leo Egghe. 2000. The distribution of n-grams. *Scientometrics*, 47(2):237–252, February.

Kuzman Ganchev and Mark Dredze. 2008. Small statistical models by random feature mixing. In *Proc. ACL 2008 Workshop on Mobile Language Processing*, pages 19–20.

Jianfeng Gao, Galen Andrew, Mark Johnson, and Kristina Toutanova. 2007. A comparative study

of parameter estimation methods for statistical natural language processing. In *Proc. ACL 2007*, pages 824–831.

Yoav Goldberg and Michael Elhadad. 2008. splitSVM: Fast, space-efficient, non-heuristic, polynomial kernel computation for NLP applications. In *Proc. ACL 2008, Short Papers*, pages 237–240.

Joshua Goodman. 2004. Exponential priors for maximum entropy models. In *Proc. HLT-NAACL 2004*, pages 305–311.

Hideki Isozaki and Hideto Kazawa. 2002. Efficient support vector classifiers for named entity recognition. In *Proc. COLING 2002*, pages 1–7.

Masakazu Iwatate, Masayuki Asahara, and Yuji Matsumoto. 2008. Japanese dependency parsing using a tournament model. In *Proc. COLING 2008*, pages 361–368.

Guy Jacobson. 1989. Space-efficient static trees and graphs. In *Proc. FOCS 1989*, pages 549–554.

Hisashi Kashima, Tsuyoshi Idé, Tsuyoshi Kato, and Masashi Sugiyama. 2009. Recent advances and trends in large-scale kernel methods. *IEICE Transactions on on Information and Systems*, E92-D. to appear.

Jun'ichi Kazama and Jun'ichi Tsujii. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *Proc. EMNLP 2003*, pages 137–144.

Jun'ichi Kazama and Jun'ichi Tsujii. 2005. Maximum entropy models with inequality constraints: A case study on text categorization. *Machine Learning*, 60(1-3):159–194.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. ACL 2008*, pages 595–603.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proc. CoNLL 2002*, pages 1–7.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proc. ACL 2003*, pages 24–31.

Sadao Kurohashi and Makoto Nagao. 2003. Building a Japanese parsed corpus. In Anne Abeillé, editor, *Treebank: Building and Using Parsed Corpora*, pages 249–260. Kluwer Academic Publishers.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML 2001*, pages 282–289.

Yudong Liu and Anoop Sarkar. 2007. Experimental evaluation of LTAG-based features for semantic role labeling. In *Proc. EMNLP 2007*, pages 590–599.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proc. HLT-NAACL 2006*, pages 152–159.

Yusuke Miyao and Jun'ichi Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proc. HLT 2002*, pages 292–297.

Ngan L.T. Nguyen and Jin-Dong Kim. 2008. Exploring domain differences for the design of a pronoun resolution system for biomedical texts. In *Proc. COLING 2008*, pages 625–632.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. IWPT 2003*, pages 149–160.

Daisuke Okanohara and Jun'ichi Tsujii. 2009. Learning combination features with L_1 regularization. In *Proc. HLT-NAACL 2009, Short Papers*, pages 97–100.

Francesco Orabona, Joseph Keshet, and Barbara Caputo. 2008. The projectron: a bounded kernel-based perceptron. In *Proc. ICML 2008*, pages 720–727.

Patrick Pantel. 2007. Data catalysis: Facilitating large-scale natural language data processing. In *Proc. ISUC*, pages 201–204.

Stijn De Saeger, Kentaro Torisawa, and Jun'ichi Kazama. 2009. Mining web-scale treebanks. In *Proc. NLP 2009*, pages 837–840.

Manabu Sassano. 2004. Linear-time dependency analysis for Japanese. In *Proc. COLING 2004*, pages 8–14.

Drahomíra "Johanka" Spoustová, Jan Hajič, Jan Raab, and Miroslav Spousta. 2009. Semi-supervised training for the averaged perceptron POS tagger. In *Proc. EACL 2009*, pages 763–771.

Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. 2004. Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data. In *Proc. ICML 2004*, pages 783–790.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*, 58(1):267–288, April.

Yu-Chieh Wu, Jie-Chi Yang, and Yue-Shi Lee. 2007. An approximate approach for training polynomial kernel SVMs in linear time. In *Proc. ACL 2007 Poster and Demo Sessions*, pages 65–68.

Susumu Yata, Kazuhiro Morita, Masao Fuketa, and Jun'ichi Aoe. 2008. Fast string matching with space-efficient word graphs. In *Proc. Innovations in Information Technology 2008*, pages 79–83.

George K. Zipf. 1949. *Human Behavior and the Principle of Least-Effort*. Addison-Wesley.

K-Best Combination of Syntactic Parsers

Hui Zhang[1, 2] **Min Zhang**[1] **Chew Lim Tan**[2] **Haizhou Li**[1]
[1]Institute for Infocomm Research [2]National University of Singapore
zhangh1982@gmail.com {mzhang, hli}@i2r.a-star.edu.sg tancl@comp.nus.edu.sg

Abstract

In this paper, we propose a linear model-based general framework to combine k-best parse outputs from multiple parsers. The proposed framework leverages on the strengths of previous system combination and re-ranking techniques in parsing by integrating them into a linear model. As a result, it is able to fully utilize both the logarithm of the probability of each k-best parse tree from each individual parser and any additional useful features. For feature weight tuning, we compare the simulated-annealing algorithm and the perceptron algorithm. Our experiments are carried out on both the Chinese and English Penn Treebank syntactic parsing task by combining two state-of-the-art parsing models, a head-driven lexicalized model and a latent-annotation-based un-lexicalized model. Experimental results show that our F-Scores of 85.45 on Chinese and 92.62 on English outperform the previously best-reported systems by 1.21 and 0.52, respectively.

1 Introduction

Statistical models have achieved great success in language parsing and obtained the state-of-the-art results in a variety of languages. In general, they can be divided into two major categories, namely lexicalized models (Collins 1997, 1999; Charniak 1997, 2000) and un-lexicalized models (Klein and Manning 2003; Matsuzaki et al. 2005; Petrov et al. 2006; Petrov and Klein 2007). In lexicalized models, word information play a key role in modeling grammar rule generation, while un-lexicalized models usually utilize latent information derived from the parse structure diversity. Although the two models are different from each other in essence, both have achieved state-of-the-art results in a variety of languages and are complementary to each other (this will be empirically verified later in this paper). Therefore, it is natural to combine the two models for better parsing performance.

Besides individual parsing models, many system combination methods for parsing have been proposed (Henderson and Brill 1999; Zeman and Žabokrtský 2005; Sagae and Lavie 2006) and promising performance improvements have been reported. In addition, parsing re-ranking (Collins 2000; Riezler et al. 2002; Charniak and Johnson 2005; Huang 2008) has also been shown to be another effective technique to improve parsing performance. This technique utilizes a bunch of linguistic features to re-rank the k-best (Huang and Chiang 2005) output on the forest level or tree level. In prior work, system combination was applied on multiple parsers while re-ranking was applied on the k-best outputs of individual parsers.

In this paper, we propose a linear model-based general framework for multiple parsers combination. The proposed framework leverages on the strengths of previous system combination and re-ranking methods and is open to any type of features. In particular, it is capable of utilizing the logarithm of the parse tree probability from each individual parser while previous combination methods are unable to use this feature since the probabilities from different parsers are not comparable. In addition, we experiment on k-best combination while previous methods are only verified on 1-best combination. Finally, we apply our method in combining outputs from both the lexicalized and un-lexicalized parsers while previous methods only carry out experiments on multiple lexicalized parsers. We also compare two learning algorithms in tuning the feature weights for the linear model.

We perform extensive experiments on the Chinese and English Penn Treebank corpus. Experimental results show that our final results, an F-Score of 92.62 on English and 85.45 on Chinese, outperform the previously best-reported systems by 0.52 point and 1.21 point, respectively. This convincingly demonstrates the effectiveness of our proposed framework. Our study also shows that the simulated-annealing algorithm (Kirkpatrick et al. 1983) is more effective

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1552–1560,
Singapore, 6-7 August 2009.

than the perceptron algorithm (Collins 2002) for feature weight tuning.

The rest of this paper is organized as follows. Section 2 briefly reviews related work. Section 3 discusses our method while section 4 presents the feature weight tuning algorithm. In Section 5, we report our experimental results and then conclude in Section 6.

2 Related Work

As discussed in the previous section, system combination and re-ranking are two techniques to improve parsing performance by post-processing parsers' k-best outputs.

Regarding the system combination study, Henderson and Brill (1999) propose two parser combination schemes, one that selects an entire tree from one of the parsers, and one that builds a new tree by selecting constituents suggested by the initial trees. According to the second scheme, it breaks each parse tree into constituents, calculates the count of each constituent, then applies the majority voting to decide which constituent would appear in the final tree. Sagae and Lavie (2006) improve this second scheme by introducing a threshold for the constituent count, and search for the tree with the largest number of count from all the possible constituent combination. Zeman and Žabokrtský (2005) study four combination techniques, including voting, stacking, unbalanced combining and switching, for constituent selection on Czech dependency parsing. Promising results have been reported in all the above three prior work. Henderson and Brill (1999) combine three parsers and obtained an F1 score of 90.6, which is better than the score of 88.6 obtained by the best individual parser as reported in their paper. Sagae and Lavie (2006) combine 5 parsers to obtain a score of 92.1, while they report a score of 91.0 for the best single parser in their paper. Finally, Zeman and Žabokrtský (2005) reports great improvements over each individual parsers and show that a parser with very low accuracy can also help to improve the performance of a highly accurate parser. However, there are two major limitations in these prior works. First, only one-best output from each individual parsers are utilized. Second, none of these works uses the parse probability of each parse tree output from the individual parser.

Regarding the parser re-ranking, Collins (2000) proposes a dozen of feature types to re-rank k-best outputs of a single head-driven parser. He uses these feature types to extract around half a million different features on the training set, and then examine two loss functions, MRF and Boosting, to do feature selection. Charniak and Johnson (2005) generate a more accurate k-best output and adopt MaxEnt method to estimate the feature weights for more than one million features extracted from the training set. Huang (2008) further improves the re-ranking work of Charniak and Johnson (2005) by re-ranking on packed forest, which could potentially incorporate exponential number of k-best list. The re-ranking techniques also achieve great improvement over the original individual parser. Collins (2002) improves the F1 score from 88.2% to 89.7%, while Charniak and Johnson (2005) improve from 90.3% to 91.4%. This latter work was then further improved by Huang (2008) to 91.7%, by utilizing the benefit of forest structure. However, one of the limitations of these techniques is the huge number of features which makes the training very expensive and inefficient in space and memory usage.

3 K-best Combination of Lexicalized and Un-Lexicalized Parsers with Model Probabilities

In this section, we first introduce our proposed k-best combination framework. Then we apply this framework to the combination of two state-of-the-art lexicalized and un-lexicalized parsers with an additional feature inspired by traditional combination techniques.

3.1 K-best Combination Framework

Our proposed framework consists of the following steps:

1) Given an input sentence and N different parsers, each parser generates K-best parse trees.
2) We combine the N*K output trees and remove any duplicates to obtain M unique tress.
3) For each of the M unique trees, we re-evaluate it with all the N models which are used by the N parsers. It is worth noting that this is the key point (i.e. one of the major advantages) of our method since some parse trees are only generated from one or I (I<N) parsers. For example, if a tree is only generated from head-driven lexicalized model, then it only has the head-driven model score. Now we re-evaluate it with the latent-annotation un-lexicalized model to reflect the latent-

annotation model's confidence for this tree. This enables our method to effectively utilize the confidence measure of all the individual models without any bias. Without this re-evaluation step, the previous combination methods are unable to utilize the various model scores.

4) Besides model scores, we also compute some additional feature scores for each tree, such as the widely-used "constituent count" feature.
5) Then we adopt the linear model to balance and combine these feature scores and generate an overall score for each parse tree.
6) Finally we re-rank the M best trees and output the one with the highest score.

$$f(t) = \lambda_1 f_1(t) + \cdots + \lambda_N f_N(t) + \lambda'_1 f'_1(t) + \cdots + \lambda'_L f'_L(t)$$

The above is the linear function used in our method, where t is the tree to be evaluated, f_1 to f_N are the model confidence scores (in this paper, we use logarithm of the parse tree probability) from the N models, λ_1 to λ_N are their weights, f'_1 to f'_L are the L additional features, λ'_1 to λ'_L are their weights.

In this paper, we employ two individual parsing model scores and only one additional feature. Let f_1 be the head-driven model score, f_2 be the latent-annotation model score, f'_1 be the constituent count feature and λ'_1 is the weight of feature f'_1.

3.2 Confidences of Lexicalized and Unlexicalized Model

The term "confidence" was used in prior parser combination studies to refer to the accuracy of each individual parser. This reflects how much we can trust the parse output of each parser. In this paper, we use the term "confidence" to refer to the logarithm of the tree probability computed by each model, which is a direct measurement of the model's confidence on the target tree being the best or correct parse output. In fact, the feature weight λ_i in our linear model functions similarly as the traditional "confidence". However, we do not directly use parser's accuracy as its value. Instead we tune it automatically on development set to optimize it against the parsing performance directly. In the following, we introduce the state-of-the-art head-driven lexicalized and latent-annotation un-lexicalized models (which are used as two individual models in this paper), and describe how they compute the tree probability briefly.

Head-driven model is one of the most representative lexicalized models. It attaches the head word to each non-terminal and views the generation of each rule as a Markov process first from father to head child, and then to the head child's left and right siblings.

Take following rule r as example,

$$F \rightarrow L_n \ldots L_2 L_1 M R_1 R_2 \ldots R_m$$

F is the rule's left hand side (i.e. father label), M is the head child, L_i is M's left sibling and R_i is M's right sibling. Let h be M's head word, the probability of this rule is

$$P(r) = P(M|h,F) \cdot \prod_{i=1}^{n} P(L_i|L_{i-1},M,h,F) \cdot \prod_{j=1}^{m} P(R_j|R_{j-1},M,h,F)$$

The probability of a tree is just the product of the probabilities of all the rules in it. The above is the general framework of head-driven model. For a specific model, there may be some additional features and modification. For example, the model2 in Collins (1999) introduces subcategorization and model3 introduces gap as additional features. Charniak (2000)'s model introduces pre-terminal as additional features.

The latent-annotation model (Matsuzaki et al. 2005; Petrov et al. 2006) is one of the most effective un-lexicalized models. Briefly speaking, latent-annotation model views each non-terminal in the Treebank as a non-terminal followed by a set of latent variables, and uses EM algorithms to automatically learn the latent variables' probability functions to maximize the probability of the given training data. Take the following binarized rule as example,

$$A \rightarrow B\ C$$

could be viewed as the set of rules

$$\{A_i \rightarrow B_j\ C_k | i,j,k\ are\ latent\ variables\}$$

The process of computing the probability of a normal tree is to first binarized all the rules in it, and then replace each rule to the corresponding set of rules with latent variables. Now the previous tree becomes a packed forest (Klein and Manning 2001; Petrov et al. 2007) in the latent-annotation model, and its probability is the inside probability of the root node. This model is quite different from the head-driven model in which

the probability of a tree is just the product all the rules' probability.

3.3 Constituent Counts

Besides the two model scores, we also adopt constituent count as an additional feature inspired by (Henderson and Brill 1999) and (Sagae and Lavie 2006). A constituent is a non-terminal node covering a special span. For example, "NP[2,4]" means a constituent labelled as "NP" which covers the span from the second word to the fourth word. If we have 100 trees and NP[2,4] appears in 60 of them, then its constituent count is 60. For each tree, its constituent count is the sum of all the counts of its constituent. However, as suggested in (Sagae and Lavie 2006), this feature favours precision over recall. To solve this issue, Sagae and Lavie (2006) use a threshold to balance them. For any constituent, we calculate its count if and only if it appears more than X times in the k-best trees; otherwise we set it as 0. In this paper, we normalize this feature by dividing the constituent count by the number of k-best. Note that the threshold value and the additional feature value are not independent. Once the threshold changes, the feature value has to be recalculated.

In conclusion, we have four parameters to estimate: two model score weights, one additional feature weight and a threshold for the additional feature.

4 Parameter Estimation

We adopt the minimum error rate principle to tune the feature weights by minimizing the error rate (i.e. maximizing the F1 score) on the development set. In our study, we implement and compare two algorithms, the simulated-annealing style algorithm and the average perceptron algorithm.

4.1 Simulated Annealing

Simulated-annealing algorithm has been proved to be a powerful and efficient algorithm in solving NP problem (Černý 1985). Fig 1 is the pseudo code of the simulated-annealing algorithm that we apply.

In a single iteration (line 4-11), the simulated algorithm selects some random points (the Markov link) for hill climbing. However, it accepts some bad points with a threshold probability controlled by the annealing temperature (line 7-10). The hill climbing nature gives this algorithm the ability of converging at local maximal point and the random nature offers it the chance to jump from some local maximal points to global maximal point. We do a slight modification to save the best parameter so far across all the finished iterations and let it be the initial point for upcoming iterations (line 12-17).

RandomNeighbour(p) is the function to generate a random neighbor for the p (the four-tuple parameter to be estimated). F1(p) is the function to calculate the F1 score over the entire test set. Given a fixed parameter p, it selects the candidate tree with best score for each sentence and computes the F1 score with the PARSEVAL metrics.

Pseudo code 1. *Simulated-annealing algorithm*
Input: *k-best trees combined from two model output*
Notation:
p: the current parameter value
F1(p): the F1 score with the parameter value p
TMF: the max F1 score of each iteration
TMp: the optimal parameter value during iteration
MaxF1: the max F1 score on dev set
Rp: the parameter value which maximizes the F1 score of the dev set
T: annealing temperature
L: length of Markov link
Output: Rp

```
1.  MaxF1:= 0, Rp:= (0,0,0,0), T:=1, L=100 // initialize
2.  Repeat                                  // iteration
3.      TMp :=Rp
4.      for  i := 1 to L  do
5.          p := RandomNeighbour(TMp)
6.          d= F1(p)- TMF
7.          if d>0 or exp(d/T) > random[0,1) then
8.              TMF:=F1(p)
9.              TMp:=p
10.         end if
11.     end for
12.     if TMF > MaxF1 then
13.         MaxF:=TMF
14.         Rp:=TMp
15.     else
16.         TMp:=Rp
17.     end if
18.     T=T*0.9
19. Until convergence
```

Fig 1. Simulated Annealing Algorithm

4.2 Averaged Perceptron

Another algorithm we apply is the averaged perceptron algorithm. Fig 2 is the pseudo code of this algorithm. Averaged perceptron is an online algorithm. It iterates through each instance. In each instance, it selects the candidate answer with the maximum function score. Then it updates the weight by the margin of feature value between the select answer and the oracle answer (line 5-9). After each iteration, it does average to generate a new weight (line 10). The averaged

perceptron has a solid theoretical fundamental and was proved to be effective across a variety of NLP tasks (Collins 2002).

However, it needs a slightly modification to adapt to our problem. Since the threshold and the constituent count are not independent, they are not linear separable. In this case, the perceptron algorithm cannot be guaranteed to converge. To solve this issue, we introduce an outer loop (line 2) to iterate through the value range of threshold with a fixed step length and in the inner loop we use perceptron to estimate the other three parameters. Finally we select the final parameter which has maximum F1 score across all the iteration (line 14-17).

Pseudo code 2. *Averaged perceptron algorithm*
Input: *k-best trees combined from two model output*
Notation:

MaxF1, Rp: already defined in pseudo code 1
T: the max number of iterations
I: the number of instances
Threshold: the threshold for constituent count
w: the three feature weights other than threshold
y': the candidate tree with max function score given a fixed weight w
y^+: the candidate tree with the max F1 score (since the oracle tree may not appeared in our candidate set, we choose this one as the pseudo orcale tree)
cand(i): the set of candidate tree for ith sentence

Output: Rp

1. MaxF1:=0, T=30
2. **for** Threshold :=0 to 1 with step 0.01 **do**
3. Initialize w
4. **for** iter : 1 to T **do**
5. **for** i := 1 to I **do**
6. $y' := \text{argmax}_{y \in \text{cand(i)}} w \cdot f(y)$
7. $w := w + f(y^+) - f(y')$
8. $w_i := w$
9. **end for**
10. $w := \frac{\sum_{i=1}^{I} w_i}{I}$
11. **if** converged **then break**
12. **end for**
13. p := (Threshold, w)
14. **if** F1(p) > MaxF1 **then**
15. MaxF1 := F1(p)
16. Rp:=p
17. **end if**
18. **end for**

Fig 2. Averaged Perceptron Algorithm

5 Experiments

We evaluate our method on both Chinese and English syntactic parsing task with the standard division on Chinese Penn Treebank Version 5.0 and WSJ English Treebank 3.0 (Marcus et al. 1993) as shown in Table 1.

We use Satoshi Sekine and Michael Collins' EVALB script modified by David Ellis for accuracy evaluation. We use Charniak's parser (Charniak 2000) and Berkeley's parser (Petrov and Klein 2007) as the two individual parsers, where Charniak's parser represents the best performance of the lexicalized model and the Berkeley's parser represents the best performance of the un-lexicalized model. We retrain both of them according to the division in Table. 1. The number of EM iteration process for Berkeley's parser is set to 5 on English and 6 on Chinese. Both the Charniak's parser and Berkeley's parser provide function to evaluate an input parse tree's probability and output the logarithm of the probability.

Lang. \ Div.	Train	Dev	Test
English	Sec.02-21	Sec. 22	Sec. 23
Chinese	Art. 001-270, 400-1151	Art. 301-325	Art. 271-300

Table 1. Data division

5.1 Effectiveness of our Combination Method

This sub-section examines the effectiveness of our proposed methods. The experiment is set up as follows: 1) for each sentence in the dev and test sets, we generate 50-best from Charniak's parser (Charniak 2000) and Berkeley's parser (Petrov and Klein 2007), respectively; 2) the two 50-best trees are merged together and duplication was removed; 3) we tune the parameters on the dev set and test on the test set. (Without specific statement, we use simulated-annealing as default weight tuning algorithm.)

The results are shown in Table 2 and Table 3. "P" means precision, "R" means recall and "F" is the F1-measure (all is in % percentage metrics); "Charniak" represents the parser of (Charniak 2000), "Berkeley" represents the parser of (Petrov and Klein 2007), "Comb." represents the combination of the two parsers.

accuracy \ parser		Charniak	Berkeley	Comb.
<=40 words	P	85.20	86.65	**90.44**
	R	83.70	84.18	**85.96**
	F	84.44	85.40	**88.15**
All	P	82.07	84.63	**87.76**
	R	79.66	81.69	**83.27**
	F	80.85	83.13	**85.45**

Table 2. Results on Chinese

parser / accuracy		Charniak	Berkeley	Comb.
<=40 words	P	90.45	90.27	**92.36**
	R	90.14	89.76	**91.42**
	F	90.30	90.02	**91.89**
All	P	89.86	89.77	**91.89**
	R	89.53	89.26	**90.97**
	F	89.70	89.51	**91.43**

Table 3. Results on English

From Table 2 and Table 3, we can see our method outperforms the single systems in all test cases with all the three evaluation metrics. Using the entire Chinese test set, our method improves the performance by 2.3 (85.45-83.13) point in F1-Score, representing 13.8% error rate reduction. Using the entire English test set, our method improves the performance by 1.7 (91.43-89.70) point in F1-Score, representing 16.5% error rate reduction. These improvements convincingly demonstrate the effectiveness of our method.

5.2 Effectiveness of K

Fig 3 and Fig. 4 show the relationship between F1 score and the number of K-best used when doing combination on Chinese and English respectively.

From Fig 3 and Fig. 4, we could see that the F1 score first increases with the increasing of K (there are some vibration points, this may due to statistical noise) and reach the peak when K is around 30-50, then it starts to drop. It shows that k-best list did provide more information than one-best and thus can help improve the accuracy; however more k-best list may also contain more noises and these noises may hurt the final combination quality.

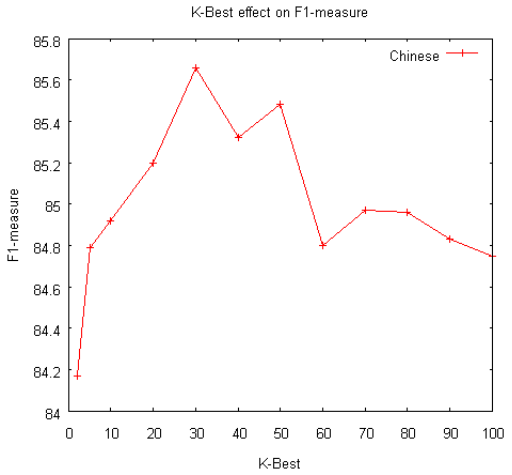


Fig 3. F1-measure vs. K on Chinese

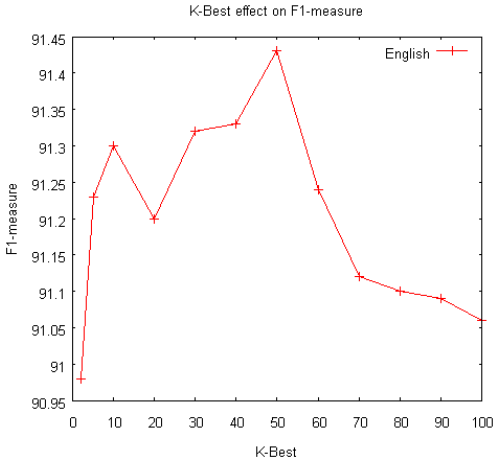


Fig 4. F1-measure vs. K on English

5.3 Diversity on the K-best Output of the Head-driven and Latent-annotation-driven Model

In this subsection, we examine how different of the 50-best trees generated from Charnriak's parser (head-driven model) (Charnriak, 2000) and Berkeley's parser (latent-annotation model) (Petrov and Klein, 2007).

Table 4 reports the statistics on the 50-best output for Chinese and English test set. Since for some short sentences the parser cannot generate up to 50 best trees, the average number of trees is less than 50 for each sentence. Each cell reports the total number of trees generated over the entire test set followed by the average count for each sentence in bracket. "Total" means simply combine the number of trees from the two parsers while "Unique" means the number after removing the duplicated trees for each sentence. In the last row, we report the averaged redundant rate for each sentence, which is derived by dividing the figures in the row "Duplicated" by those in the row "Total".

	Chinese	English
Charniak	14577 (41.9)	120438 (49.9)
Berkeley	14524 (41.7)	114299 (47.3)
Total	29101 (83.6)	234737 (97.2)
Unique	27747 (79.7)	221633 (91.7)
Duplicated	1354 (3.9)	13104 (5.4)
Redundant rate	4.65%	5.58%

Table 4. The statistics on the 50-best output for Chinese and English test set.

The small redundant rate clearly suggests that the two parsing models are quite different and are complementary to each other.

Oracle \ parser		Charniak	Berkeley	Comb.
Chinese	P	88.95	90.07	**92.45**
	R	86.51	87.12	**89.67**
	F	87.71	88.57	**91.03**
English	P	97.06	95.86	**98.10**
	R	96.57	95.53	**97.68**
	F	96.82	95.70	**97.89**

Table 5. The oracle over 50-best output for individual parser and our method

The k-best oracle score is the upper bound of the quality of the k-best trees. Table 5 reports the oracle score for the 50-best of the two individual parsers and our method. Similar to Table 4, Table 5 shows again that the two models are complementary to each other and our method is able to take the strength of the two models.

5.4 Effectiveness of Model Confidence

One of the advantages of our method that we claim is that we can utilize the feature of the model confidence score (logarithm of the parse tree probability).

Table 6 shows that all the three features contribute to the final accuracy improvement. Even if we only use the "B+C" confidence scores, it also outperforms the baseline individual parser (as reported in Table 2 and Table 3) greatly. All these together clearly verify the effective of the model confidence feature and our method can effectively utilize this feature.

Lang \ Feat.	I	B+C	B+C+I
Chinese	82.34	84.67	**85.45**
English	90.20	91.02	**91.43**

Table 6. F1 score on 50-best combination with different feature configuration. "I" means the constituent count, "B" means Berkeley parser confidence score and "C" means Charniak parser confidence score.

5.5 Comparison of the Weight Tuning Algorithms

In this sub-section, we compare the two weight tuning algorithms on 50-best combination tasks on both Chinese and English. Dan Bikel's randomized parsing evaluation comparator (Bikel 2004) was used to do significant test on precision and recall metrics. The results are shown in Table 7.

We can see, simulated annealing outperforms the averaged perceptron significantly in both precision ($p<0.005$) and recall ($p<0.05$) metrics of Chinese task and precision ($p<0.005$) metric of English task. Though averaged perceptron got slightly better recall score on English task, it is not significant according to the p-value ($p>0.2$).

From table 8, we could see the simulated annealing algorithm is around 2-4 times slower than averaged perceptron algorithm.

Lang \ Algo.		SA.	AP.	P-value
Chinese	P	**87.76**	86.85	0.003
	R	**83.27**	82.90	0.030
English	P	**91.89**	91.72	0.004
	R	90.97	**91.02**	0.205

Table 7. Precision and Recall score on 50-best combination by the two parameter estimation algorithms with significant test; "SA." is simulated annealing, "AP." is averaged perceptron, "P-value" is the significant test p-value.

Lang \ Algo.	Simulated Annealing	Averaged Perceptron
Chinese	2.3	0.6
English	12	6

Table 8. Time taken (in minutes) on 50-best combination of the two parameter estimation algorithms

5.6 Performance-Enhanced Individual Parsers on English

For Charniak's lexicalized parser, there are two techniques to improve its performance. One is reranking as explained in section 2. The other is the self-training (McClosky et al. 2006) which first parses and reranks the NANC corpus, and then use them as additional training data to retrain the model. In this sub-section, we apply our method to combine the Berkeley parser and the enhanced Charniak parser by using the new model confidence score output from the enhanced Charniak parser.

Table 9 and Table 10 show that the Charniak parser enhanced by re-ranking and self-training is able to help to further improve the performance of our method. This is because that the enhanced Charniak parser provides more accurate model confidence score.

parser / accuracy		reranking	Comb.	baseline
<=40 words	P	92.34	**93.41**	92.36
	R	91.61	**92.15**	91.42
	F	91.97	**92.77**	91.89
All	P	91.78	**92.92**	91.89
	R	91.03	**91.70**	90.97
	F	91.40	**92.30**	91.43

Table 9. Performance with Charniak parser enhanced by re-ranking; "baseline" is the performance of the combination of Table 3.

parser / accuracy		self-train+ reranking	Comb.	baseline
<=40 words	P	92.87	**93.69**	92.36
	R	92.12	**92.44**	91.42
	F	92.49	**93.06**	91.89
All	P	92.41	**93.25**	91.89
	R	91.64	**92.00**	90.97
	F	92.02	**92.62**	91.43

Table 10. Performance with Charniak parser enhanced by re-ranking plus self-training

5.7 Comparison with Other State-of-the-art Results

Table 11 and table 12 compare our method with the other state-of-the-art methods; we use I, B, R, S and C to denote individual model (Charniak 2000; Collins 2000; Bod 2003; Petrov and Klein 2007), bilingual-constrained model (Burkett and Klein 2008)[1], re-ranking model (Charniak and Johnson 2005, Huang 2008), self-training model (David McClosky 2006) and combination model (Sagae and Lavie 2006) respectively. The two tables clearly show that our method advance the state-of-the-art results on both Chinese and English syntax parsing.

System		F1-Measure
I	Charniak (2000)	80.85
	Petrov and Klein (2007)	83.13
B	Burkett and Klein (2008)[1]	84.24
C	**Our method**	**85.45**

Table 11. Accuracy comparison on Chinese

System		F1-Measure
I	Petrov and Klein (2007)	89.5
	Charniak (2000)	89.7
	Bod (2003)	90.7
R	Collins (2000)	89.7
	Charniak and Johnson (2005)	91.4
	Huang (2008)	91.7
S	David McClosky (2006)	92.1
C	Sagae and Lavie (2006)	92.1
	Our method	**92.6**

Table 12. Accuracy comparison on English.

6 Conclusions

In this paper, we propose a linear model-based general framework for multiple parser combination. Compared with previous methods, our method is able to use diverse features, including logarithm of the parse tree probability calculated by the individual systems. We verify our method by combining the two representative parsing models, lexicalized model and un-lexicalized model, on both Chinese and English. Experimental results show our method is very effective and advance the state-of-the-art results on both Chinese and English syntax parsing. In the future, we will explore more features and study the forest-based combination methods for syntactic parsing.

Acknowledgement

We would like to thank Prof. Hwee Tou Ng for his help and support; Prof. Charniak for his suggestion on doing the experiments with the self-trained parser and David McCloksy for his help on the self-trained model; Yee Seng Chan and the anonymous reviewers for their valuable comments.

References

Dan Bikel. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models.* Ph.D. Thesis, University of Pennsylvania 2004.

Rens Bod. 2003. *An efficient implementation of a new DOP model*. EACL-04.

David Burkett and Dan Klein. 2008. *Two Languages are Better than One (for Syntactic Parsing).* EMNLP-08.

[1] Burkett and Klein (2008) use the additional knowledge from Chinese-English parallel Treebank to improve Chinese parsing accuracy.

The corresponding authors of this paper are Hui Zhang (zhangh1982@gmail.com) and Min Zhang (mzhang@i2r.a-star.edu.sg)

V Černý 1985. *Thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm.* Journal of Optimization Theory and Applications, 45:41-51.1985.

Eugene Charniak. 1997. *Statistical parsing with a context-free grammar and word statistics.* AAAI-97, pages 598-603.

Eugene Charniak. 2000. *A maximum-entropy-inspired parser.* NAACL-2000.

Eugene Charniak and Mark Johnson. 2005. *Coarse-to-fine-grained n-best parsing and discriminative reranking.* ACL-05, Pages 173-180.

Michael Collins. 1997. *Three generative, lexicalised models for statistical parsing.* ACL-97, pages 16-23.

Michael Collins.1999. *Head-drivenstatistical models for natural language parsing.* Doctoral Dissertation, Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia 1999.

Michael Collins. 2000. *Discriminative reranking for natural language parsing.* ICML-00, pages 175-182.

Michael Collins. 2002. *Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms.* EMNLP-02.

Liang Huang. 2008. *Forest Reranking: Discriminative Parsing with Non-Local Features.* ACL-HLT-08, pages 586-594.

Liang Huang and David Chiang. 2005. *Better k-best Parsing.* IWPT-05.

S. Kirkpatrick, C. D. Gelatt, Jr. and M. P. Vecchi. 1983. *Optimization by Simulated Annealing*. Science. New Series 220 (4598): 671-680.

Dan Klein and Christopher D. Manning. 2001. *Parsing and Hypergraphs.* IWPT-01.

Dan Klein and Christopher D. Manning. 2003. *Accurate unlexicalized parsing.* ACL-03, pages 423-430.

John Henderson and Eric Brill. 1999. *Exploiting diversity in natural language processing: combining parsers.* EMNLP-99.

Mitchell P. Marcus, Beatrice Santorini and Mary Ann Marcinkiewicz. 1993. *Building a large annotated corpus of English: the Penn Treebank.* Computational Linguistics, 19:313-330.

Takuya Matsuzaki. Yusuke Miyao and Jun'ichi Tsujii. 2005. *Probabilistic CFG with latent annotations.* ACL-05, pages 75-82.

David McClosky, Eugene Charniak and Mark Johnson. 2006. *Effective self-training for parsing.* NAACL-06, pages 152-159.

Slav Petrov, Leon Barrett, Romain Thibaux and Dan Klein. 2006. *Learning accurate, compact, and interpretable tree annotation.* COLING-ACL-06, pages 443-440.

Slav Petrov and Dan Klein. 2007. *Improved inference for unlexicalized parsing.* HLT-NAACL-07, pages 401-411.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. III Maxwell and Mark Johnson. 2002. *Parsing the wall street journal using a lexical-functional grammar and discriminative estimation techniques.* ACL-02, pages 271–278.

Kenji Sagae and Alon Lavie. 2006. *Parser combination by reparsing*. HLT-NAACL-06, pages 129-132.

Daniel Zeman and Zdeněk Žabokrtský. *Improving Parsing Accuracy by Combining Diverse Dependency Parsers.* IWPT-05.

Chinese Novelty Mining

Yi Zhang
Nanyang Technological University
50 Nanyang Avenue
Singapore 639798
yizhang@ntu.edu.sg

Flora S. Tsai
Nanyang Technological University
50 Nanyang Avenue
Singapore 639798
fst1@columbia.edu

Abstract

Automated mining of novel documents or sentences from chronologically ordered documents or sentences is an open challenge in text mining. In this paper, we describe the preprocessing techniques for detecting novel Chinese text and discuss the influence of different Part of Speech (POS) filtering rules on the detection performance. Experimental results on AP-WSJ and TREC 2004 Novelty Track data show that the Chinese novelty mining performance is quite different when choosing two dissimilar POS filtering rules. Thus, the selection of words to represent Chinese text is of vital importance to the success of the Chinese novelty mining. Moreover, we compare the Chinese novelty mining performance with that of English and investigate the impact of preprocessing steps on detecting novel Chinese text, which will be very helpful for developing a Chinese novelty mining system.

1 Introduction

The bloom of information nowadays brings us rich useful information as well as tons of redundant information in news articles, social networks (Tsai et al., 2009), and blogs (Chen et al., 2008). Novelty mining (NM), or novelty detection, aims at mining novel information from a chronologically ordered list of relevant documents/sentences. It can facilitate users to quickly get useful information without going through a lot of redundant information, which is usually a tedious and time-consuming task.

The process of detecting novel text contains three main steps, (i) preprocessing, (ii) categorization, and (iii) novelty mining. The first step preprocesses the text documents/sentences by removing stop words, performing word stemming, implementing POS tagging etc. Categorization classifies each incoming document/sentence into its relevant topic bin. Then, within each topic bin containing a group of relevant documents/sentences, novelty mining searches through the time sequence of documents/sentences and retrieves only those with "novel" information. This paper focuses on applying document/sentence-level novelty mining on Chinese. In this task, we need to identify all novel Chinese text given groups of relevant documents/sentences.

Novelty mining has been performed at three different levels: event level, sentence level and document level (Li and Croft, 2005). Works on novelty mining at the event level originated from research on Topic Detection and Tracking (TDT), which is concerned with online new event detection/first story detection (Allan et al., 1998; Yang et al., 2002; Stokes and Carthy, 2001; Franz et al., 2001; Brants et al., 2003). Research on document and sentence-level novelty mining aims to find relevant and novel documents/sentences given a stream of documents/sentences. Previous studies on document and sentence-level novelty mining tend to apply some promising content-oriented techniques (Li and Croft, 2005; Allan et al., 1998; Yang et al., 1998; Zhang and Tsai, 2009). Similarity metrics that can be used for detecting novel text are word overlap, cosine similarity (Yang et al., 1998), new word count (Brants et al., 2003), etc. Other works utilize ontological knowledge, especially taxonomy, such as WordNet (Zhang et al., 2002; Allan et al., 2003), synonym dictionary (Franz et al., 2001), HowNet (Eichmann and Srinivasan, 2002), etc.

Previous studies for novelty mining have been conducted on the English and Malay languages (Kwee et al., 2009; Tang et al., 2009; Tang and Tsai, 2009). Novelty mining studies on the Chinese language have been performed on topic de-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1561–1570,
Singapore, 6-7 August 2009.

tection and tracking, which identifies and collects relevant stories on certain topics from information stream (Zheng et al., 2008; Hong et al., 2008). Also many works have discussed the issues, such as word segmentation, POS tagging etc, between English and Chinese (Wang et al., 2006; Wu et al., 2003). However, to the best of our knowledge, no studies have been reported on discussing preprocessing techniques on Chinese document and sentence-level novelty mining, which is the focus of our paper.

The rest of this paper is organized as follows. Section 2 gives a brief overview of related work on detecting novel documents and sentences on English and Chinese. Section 3 introduces the details of preprocessing steps for English and Chinese. A general novelty mining algorithm is described in Section 4. Section 5 reports experimental results. Section 6 summarizes the research findings and discusses issues for further research.

2 Related Work

In the pioneering work for detecting novel documents (Zhang et al., 2002), document novelty was predicted based on the distance between the new document and the previously delivered documents in history. The detected document which is very similar to any of its history documents is regarded as a redundant document. To serve users better, it could be more helpful to further highlight novel information at the sentence level. Therefore, later studies focused on detecting novel sentences, such as those reported in TREC 2002-2004 Novelty Tracks (Harman, 2002; Soboroff and Harman, 2003; Soboroff, 2004), which compared various novelty metrics (Allan et al., 2003), and integrated different natural language techniques (Ng et al., 2007; Li and Croft, 2008).

Although novelty mining studies have mainly been conducted on the English language, studies on the Chinese language have been performed on topic detection and tracking. A prior study (Zheng et al., 2008) proposed an improved relevance model to detect the novelty information in topic tracking feedback and modified the topic model based on this information. Experimental results on Chinese datasets TDT4 and TDT2003 proved the effectiveness in topic tracking. Another study proposed a method of applying semantic domain language model to link detection, based on the structure relation among contents and the semantic distribution in a story (Hong et al., 2008).

3 Preprocessing for English and Chinese

3.1 English

Since the focus of this paper is on novelty mining, we begin from a list of relevant documents or sentences that have already undergone the categorization process.

The first step for English preprocessing is to remove all stop words from documents or sentences, such as conjunctions, prepositions, and articles. Stop words are words that are too common to be informative. These words should be removed, otherwise it will influence the novelty prediction of documents or sentences. After stop words removal, the remaining words are then stemmed. The inflected (or sometimes derived) words are reduced to their root forms. This paper used Porter stemming algorithm (Porter, 1997) for English word stemming. This algorithm removes the commoner morphological and inflexional endings from the words in English. The entire preprocessing steps in English novelty mining can be seen in Figure 1.

3.2 Chinese

In Chinese, the word is the smallest independent meaningful element. There is no obvious boundary between words so that Chinese lexical analysis, such as Chinese word segmentation, is the prerequisite for novelty mining.

Unlike English, Chinese word segmentation is a very challenging problem because of the difficulties in defining what constitutes a word (Gao et al., 2005). While each criteria provides valuable insights into "word-hood" in Chinese, they do not consistently lead us to the same conclusions. Moreover, there is no white space between Chinese words or expressions and there are many ambiguities in the Chinese language, such as: '主板和服务器' (means 'mainboard and server' in English) might be '主板/和/服务器' (means 'mainboard/and/server' in English) or '主板/和服/务/器' (means 'mainboard/kimono/task/utensil' in English). This ambiguity is a great challenge for Chinese word segmentation. In addition, there is no obvious inflected or derived words in Chinese so that word stemming is not applicable.

Therefore, in order to reduce the noise brought by Chinese word segmentation and get a better

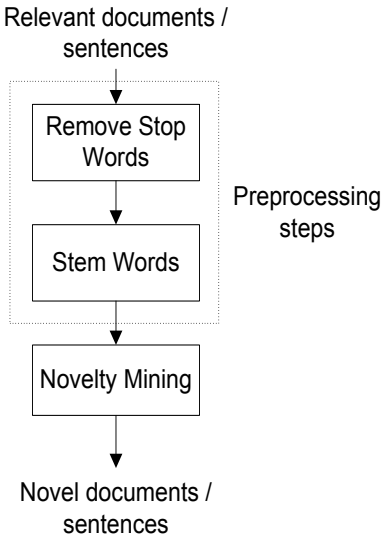


Figure 1: Preprocessing steps on English.

word list for one document or sentence, we firstly apply word segmentation on the Chinese text and then utilize Part-of-Speech (POS) tagging to select the meaningful candidate words. Figure 2 shows the preprocessing steps on the Chinese text for novelty mining. POS tagging is a process of marking up the word in a text as corresponding to a particular part of speech. It is learnt that the idea of a text mainly relies on some meaningful words, such as nouns and verbs, so that we can get the main content by extracting these meaningful words. Moreover, it will decrease the impact of the errors in Chinese word segmentation on novelty mining because only meaningful words are considered and other words (including stop words) such as '虽然' (means 'although' in English) will not appear in the word list for the following similarity computation in novelty mining. Losee also mentioned that POS tagging shows a great potential to avoid lexical ambiguity and it can help to improve the performance of information retrieval (Losee, 2001).

ICTCLAS is used when performing word segmentation and POS tagging in our experiments (ICTCLAS, 2008). It is an open source project and achieves a better precision in Chinese word segmentation and POS tagging than other Chinese POS tagging softwares (ICTCLAS, 2008). First, we apply word segmentation on the relevant Chinese documents/sentences. Chinese word segmentation includes atom segmentation, N-shortest path based rough segmentation and unknown words recognition (see Figure 3). Atom segmentation is an initial step of the Chinese language segmentation process, where atom is defined to be the minimal unit that cannot be split further. The atom can be a Chinese character, punctuation, symbol string, etc. Then, rough segmentation tries to discover the correct segmentation with as few candidates as possible. The N-Shortest Path (NSP) method (Zhang and Liu, 2002) is applied for rough segmentation. Next, we detect some unknown words such as person name, location name so as to optimize the segmentation result. Finally, we POS tag the words and keep some kinds of words in the word list according to the selective rule, which are used in novelty mining.

4 Novelty Mining

From the output of preprocessing, we can obtain a bag of words. The corresponding term-document matrix (TDM)/term-sentence matrix (TSM) can be constructed by counting the term frequency (TF) of each word. The novelty mining system predicts any incoming document/sentence by comparing it with its history documents/sentences in this vector space. Therefore, given a Chinese TDM/TSM, the novelty mining system designed for English can also be applied to Chinese.

In novelty mining, the novelty of a document/sentence can be quantitatively measured by a novelty metric and represented by a novelty score. The most popular novelty metric, i.e. cosine similarity (see (Allan et al., 2003)), is adopted. This metric first calculates the similarities between the current document/sentence d_t and each of its his-

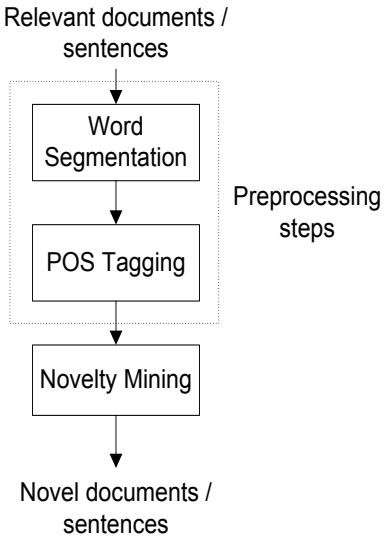


Figure 2: Preprocessing steps on Chinese.

tory documents/sentences d_i ($1 \leq i \leq t-1$). Then, the novelty score is simply one minus the maximum of these cosine similarities, as shown in Eq.(1).

$$Novelty\ Score(d_t) = 1 - \max_{1 \leq i \leq t-1} cos(d_t, d_i) \quad (1)$$

$$cos(d_t, d_i) = \frac{\sum_{k=1}^{n} w_k(d_t) \cdot w_k(d_i)}{\|d_t\| \cdot \|d_i\|}$$

where $N_{cos}(d)$ denotes the cosine similarity score of the document/sentence d and $w_k(d)$ is the weight of k^{th} element in the document/sentence weighted vector d. The term weighting function used in our work is TF(term frequency).

The final decision on whether a document/sentence is novel or not depends on whether the novelty score falls above or below a threshold. The document/sentence predicted as "novel" will be placed into the list of history documents/sentences.

5 Experiments and Results

5.1 Datasets

Two public datasets APWSJ (Zhang et al., 2002) and TREC Novelty Track 2004 (Soboroff, 2004) are selected as our experimental datasets for the document-level and the sentence-level novelty mining respectively. APWSJ data consists of news articles from Associated Press (AP) and Wall Street Journal (WSJ). There are 50 topics from Q101 to Q150 in APWSJ and 5 topics (Q131, Q142, Q145, Q147, Q150) are excluded from the experiments because they lack non-novel documents (Zhao et al., 2006). The assessors provide two degrees of judgements on non-novel documents, *absolute redundant* and *somewhat redundant*. In our experiments, we adopt the strict definition used in (Zhang et al., 2002) where only *absolute redundant* documents are regarded as non-novel. TREC 2004 Novelty Track data is developed from AQUAINT collection. Both relevant and novel sentences are selected by TREC's assessors. The statistics of these two datasets are summarized in Table 1.

Table 1: Statistics of experimental data

Dataset	Novel	Non-novel
APWSJ	10839(91.10%)	1057(8.90%)
TREC2004	3454(41.40%)	4889(58.60%)

5.2 Evaluation Measures

From many previous works, redundancy precision (RP), redundancy recall (RR) and redundancy F Score (RF) are used to evaluate the performance of document-level novelty mining (Zhang et al., 2002). Precision (P), recall (R) and F Score (F) are mainly used in evaluating the performance for sentence-level novelty mining (Allan et al., 2003). Therefore, we use RP, RR, RF and redundancy precision-recall (R-PR) curve to evaluate our experimental results on the document level. P, R, F and precision-recall (PR) curve are used to evaluate the performance on the sentence-level novelty mining. The larger the area under the R-PR

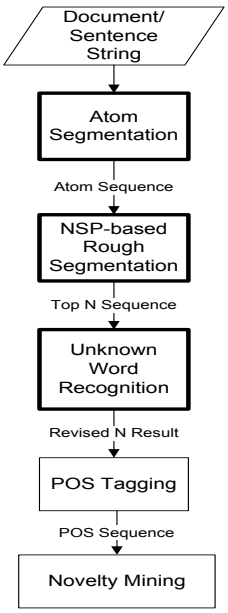


Figure 3: Word segmentation on Chinese.

curve/PR curve, the better the algorithm. Also we drew the standard redundancy F Score/F Score contours (Soboroff, 2004), which indicate the F Score values when setting precision and recall from 0 to 1 with a step of 0.1. These contours can facilitate us to compare redundancy F Scores/F Scores in R-PR curves/PR curves. Redundancy precision, redundancy recall, precision and recall on a certain topic are defined as:

$$Redundancy\ Precision = \frac{R^-}{R^- + N^-} \quad (2)$$

$$Redundancy\ Recall = \frac{R^-}{R^- + R^+} \quad (3)$$

$$Precision = \frac{N^+}{N^+ + R^+} \quad (4)$$

$$Recall = \frac{N^+}{N^+ + N^-} \quad (5)$$

where R^+,R^-,N^+,N^- correspond to the number of documents/sentences that fall into each category (see Table 2).

Based on all the topics' RP/P and RR/R, we could get the average RP/P and average RR/R by calculating the arithmetic mean of these scores on all topics. Then, the average redundancy F Score (RF)/F Score (F) is obtained by the harmonic average of the average RP/P and average RR/R.

Table 2: Categories for evaluation

	Non-novel	Novel
Delivered	R^+	N^+
Not Delivered	R^-	N^-

5.3 Experimental Results

In this experimental study, the focus was novelty mining rather than relevant documents/sentences categorization. Therefore, our experiments started with all given relevant Chinese text, from which the novel text should be identified.

Since the datasets that we used for document-level novelty mining and sentence-level novelty mining both were written in English, we first translated them into Chinese. During this process, we investigated issues on machine translation vs. manually corrected translation.

We compared the novelty mining performance on 107 text in TREC 2004 Novelty Track between automatically translated using Google Translate API[1] and the manually corrected translation. For example, here is an English sentence in Topic 51:

According to a Chilean government report, a total of 4,299 political opponents died or disappeared during Pinochet's term.

After machine translation using Google Translator, the above sentence is translated as:

根据智利政府的报告，共有4299政敌的死亡或失踪期间，皮诺切特的任期。

[1]http://code.google.com/p/google-api-translate-java

Then we manually corrected the machine translation and obtained the corrected translation:

根据智利政府的报告，在皮诺切特的任期期间，共有4299政敌死亡或失踪。

After novelty mining on the machine translation sentences and the humanly corrected translation sentences individually, we found that there is a slight difference (<2%) in precision and F Score. Thus, we used machine translation to translate the remaining documents/sentences to Chinese. This indicates that the noise in machine translation for Chinese had little impact on our actual results.

Then on English text, we applied the preprocessing steps discussed in Section 3.1, including stop word removing and word stemming. For Chinese datasets, we segmented the documents/sentences into words and then performed POS filtering to acquire the candidate words for the space vector.

Based on the vectors of Chinese text, we calculated the similarities between documents/sentences and predicted the novelty for each document/sentence in the Chinese and English datasets. An incoming Chinese/English document will be compared with all the system delivered 10 novel documents. If the novelty score is above the novelty score threshold, the document is considered to be novel. Thresholds used were between 0.05 and 0.65. We also performed Chinese/English sentence-level novelty mining. Whether an incoming Chinese/English sentence is novel is predicted by comparing with the most recent system-delivered 1000 novel sentences. Thresholds adopted were between 0.05 and 0.95 with an equal step of 0.10. Then, we evaluated the Chinese/English novel text detection performance by setting a series of novelty score thresholds.

5.3.1 POS Filtering Rule

We adopted two different rules to select the candidate words to represent one document/sentence and investigated the POS filtering influence on detecting the novel Chinese text.

- Rule1: only some non-meaningful words, including pronouns ('r' in Peking University/Chinese Academy of Sciences Chinese POS tagging criterions (PKU and CAS, 1999)), auxiliary words ('u'), tone words ('y'), conjunctions ('c'), prepositions ('p') and punctuation words ('w') are removed.
- Rule2: fewer kinds of words are selected to represent a document/sentence. Only nouns (including 'n' short for common nouns, 'nr' short for person name, 'ns' short for location name, 'nt' short for organization name, 'nz' short for other proper nouns), verbs ('v'), adjectives ('a') and adverbs ('d') are kept.

For example, here is a simple Chinese sentence: "墙上挂着一幅画。" (There is a picture on the wall). After POS filtering using Rule1, the words we keep are: "墙('n'), 上('v'), 挂('v'), 一('m'), 幅('q'), 画('n')". After POS filtering using Rule2, the remaining words are: "墙('n'), 上('v'), 挂('v'), 画('n')". It is noticed that by using Rule2, we can remove more non-important words.

Figure 4 and Figure 5 show the performances on the document and sentence-level novelty mining when choosing the stricter rule (Rule2) and the less strict rule (Rule1) in POS filtering. The grey dashed lines show contours at intervals of 0.1 points of F Score.

From Figure 4 and Figure 5, we learn that the Chinese novelty mining performance varies when choosing the stricter rule (Rule2) and the less strict rule (Rule1) in POS filtering. We can obtain a better performance when choosing a stricter rule (Rule2). Therefore, it is necessary to perform POS filtering in the preprocessing steps on Chinese and just removing some non-meaningful words (like stop words) may not be enough. POS filtering can help to remove the less meaningful words so that each vector is represented better. Compared to choosing more kinds of words (Rule1), only keeping nouns, verbs, adjectives and adverbs (Rule2) will be a better choice for novelty mining. We also noticed that the selection of words to represent Chinese text is of vital importance to the success of Chinese novelty mining.

5.3.2 Comparison with English

We compared the novelty mining performance on the English and Chinese documents/sentences datasets. For Chinese, we chose Rule2 to select the candidate words. Figure 6 and Figure 7 show the R-PR and PR curves of document/sentence-level novelty mining in English and Chinese when given a series of novelty score thresholds.

From Figure 6 and Figure 7, we observe that the performance on detecting novel Chinese documents is slightly lower than that on English. This may be due to the different linguistical characteris-

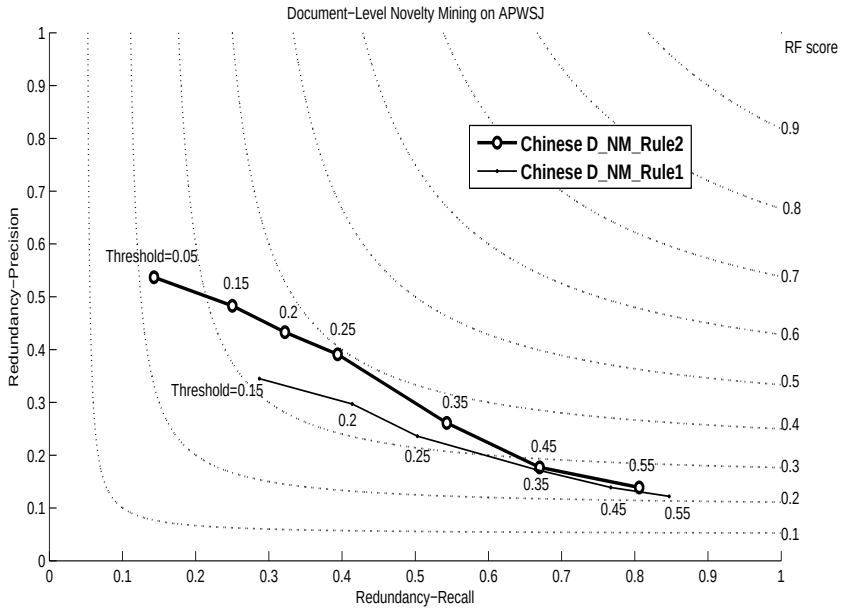


Figure 4: R-PR curves for document-level novelty mining on Chinese when choosing different rules on APWSJ. The grey dashed lines show contours at intervals of 0.1 points of RF.

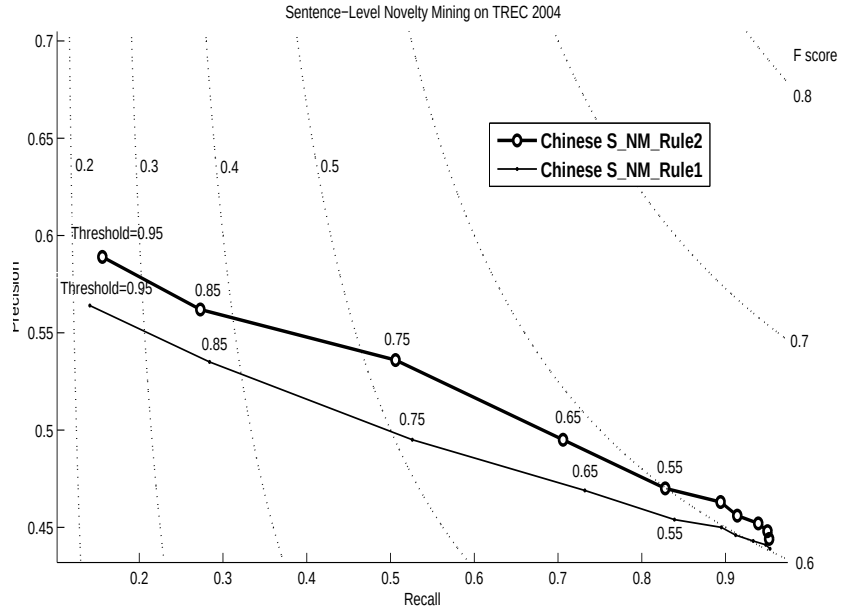


Figure 5: PR curves for sentence-level novelty mining on Chinese when choosing different rules on TREC 2004. The grey dashed lines show contours at intervals of 0.1 points of F.

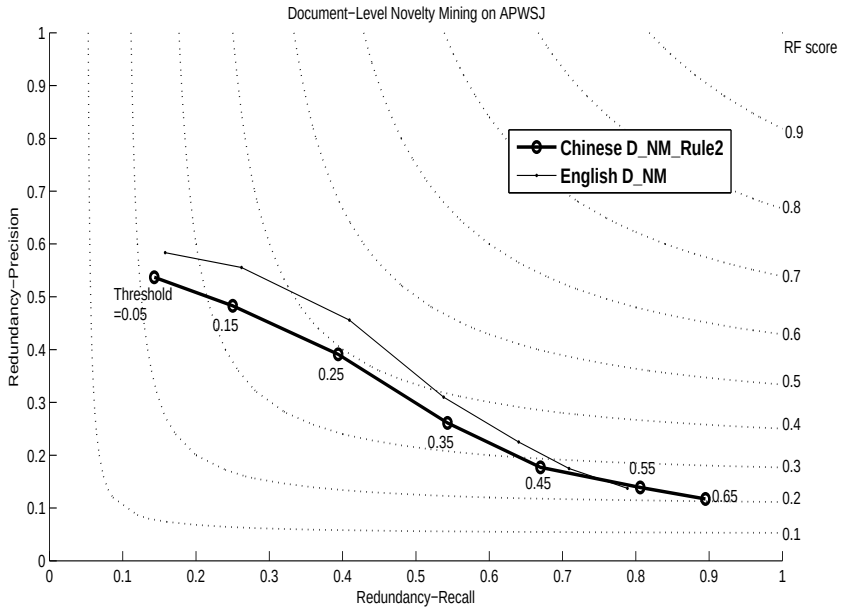


Figure 6: R-PR curves for document-level novelty mining on Chinese and English on APWSJ. The grey dashed lines show contours at intervals of 0.1 points of RF.

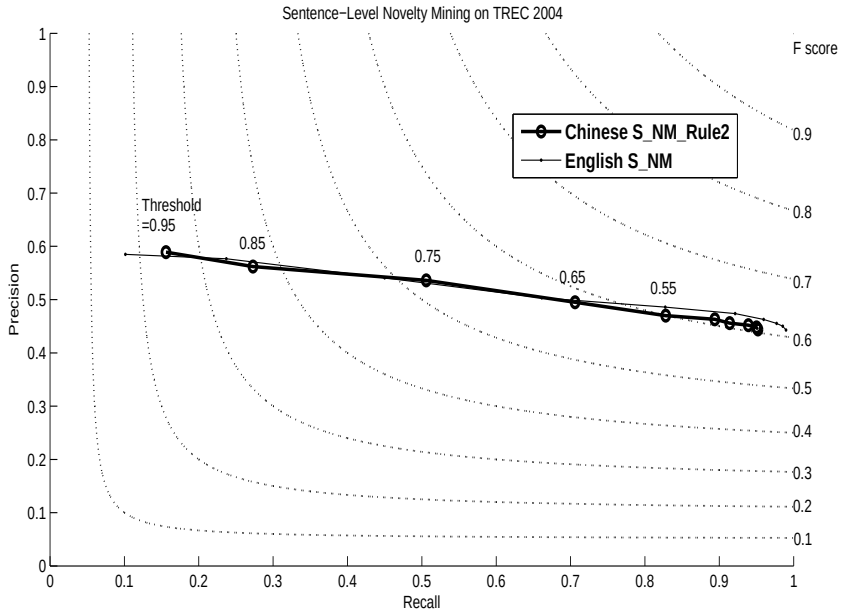


Figure 7: PR curves for sentence-level novelty mining on Chinese and English on TREC 2004. The grey dashed lines show contours at intervals of 0.1 points of F.

tics of each language so that the preprocessing influence on each language's novelty mining is dissimilar. Furthermore, the Chinese preprocessing quality is not as good as that on English so that it is difficult to obtain a good "bag of words" from a document. Moreover, the errors in word segmentation will influence the result of POS tagging. These issues make tokenizing and POS tagging extremely difficult for the Chinese text.

However, the performance of Chinese sentence-level novelty mining is almost the same as that on English. The reason is that the novelty mining performance at the sentence level is not so sensitive to the preprocessing steps as that at the document level. If the similarity computation is based on the sentence level, the word segmentation and POS tagging errors actually will not have a big influence on the result as that on documents.

6 Conclusion

This paper studied the preprocessing issues on mining novel Chinese text, which, to the best of our knowledge, have not been sufficiently addressed in previous studies. We described the Chinese preprocessing steps and discussed the influence when choosing different Part-of-Speech (POS) filtering rules. Then we applied novelty mining on Chinese and English documents/sentences and compared their performance.

The experimental results on APWSJ and TREC 2004 Novelty Track showed that after adopting a stricter POS filtering rule, the Chinese novelty mining performed better on both documents and sentences. This is because non-meaningful words have a negative influence on detecting novel text. However, compared to English, Chinese performed worse on the document level and similarly on the sentence level. The reason may be due to the lower sensitivity of preprocessing at the sentence level. The main contributions of this work are as follows:

1) We investigated the preprocessing techniques for detecting novel Chinese text on both document and sentence level.

2) The POS filtering rule, telling how to select words to represent one document/sentence, was discussed.

3) Several experiments were conducted to compare the novelty mining performance between Chinese and English. The novelty mining performance on Chinese can be improved as good as that on English if we can increase the preprocessing precision on Chinese text.

Our findings will be very helpful for developing a real-time Chinese novelty mining system at both the document and sentence level. In future work, we will try other word combinations and investigate better ways to represent the Chinese text. In addition, we will explore how to utilize the better Chinese sentence-level novelty mining result to improve the detection performance on documents.

References

James Allan, Ron Papka, and Victor Lavrenko. 1998. Online new event detection and tracking. In *SIGIR 1998, Melbourne, Australia*, pages 37–45.

James Allan, Courtney Wade, and Alvaro Bolivar. 2003. Retrieval and novelty detection at the sentence level. In *SIGIR 2003, Toronto, Canada*, pages 314–321. ACM, August.

Thorsten Brants, Francine Chen, and Ayman Farahat. 2003. A system for new event detection. In *SIGIR 2003, Toronto, Canada*, pages 330–337.

Yun Chen, Flora S. Tsai, and Kap Luk Chan. 2008. Machine learning techniques for business blog search and mining. *Expert Syst. Appl.*, 35(3):581–590.

D. Eichmann and P. Srinivasan. 2002. Novel results and some answers. In *TREC 2002 - the 11th Text REtrieval Conference*.

Martin Franz, Abraham Ittycheriah, J.Scott McCarley, and Todd Ward. 2001. First story detection: combining similarity and novelty based approach. In *Topic Detection and Tracking Workshop*.

Jianfeng Gao, Mu Li, Andi Wu, and Chang-Ning Huang. 2005. Chinese word segmentation and named entity recognition: A pragmatic approach. *Computational Linguistics*, 31(4):531–574, December.

D. Harman. 2002. Overview of the TREC 2002 Novelty Track. In *TREC 2002 - the 11th Text Retrieval Conference*, pages 46–55.

Yu Hong, Yu Zhang, Jili Fan, Ting Liu, and Sheng Li. 2008. Chinese topic link detection based on semantic domain language model. *Journal of Software*, 19(9):2265–2275.

ICTCLAS. 2008. http://ictclas.org/index.html.

Agus Trisnajaya Kwee, Flora S Tsai, and Wenyin Tang. 2009. Sentence-level novelty detection in English and Malay. In *Lecture Notes in Computer Science (LNCS)*, volume 5476, pages 40–51.

Xiaoyong Li and W. Bruce Croft. 2005. Novelty detection based on sentence level patterns. In *CIKM 2005*, pages 744–751.

Xiaoyong Li and W. Bruce Croft. 2008. An information-pattern-based approach to novelty detection. *Information Processing and Management: an International Journal*, 44(3):1159–1188, May.

Robert M. Losee. 2001. Natural language processing in support of decision making: Phrases and part-of-speech tagging. *Information Processing and Management: an International Journal*, 37(6).

Kok Wah Ng, Flora S. Tsai, Kiat Chong Goh, and Lihui Chen. 2007. Novelty detection for text documents using named entity recognition. In *Information, Communications and Signal Processing, 2007 6th International Conference on*, pages 1–5, December.

PKU and CAS. 1999. Chinese POS tagging criterion. http://icl.pku.edu.cn/icl_groups/corpus/addition.htm.

M.F. Porter. 1997. An algorithm for suffix stripping. *Readings in information retrieval*, pages 313–316.

Ian Soboroff and D. Harman. 2003. Overview of the TREC 2003 Novelty Track. In *TREC 2003 - the 12th Text Retrieval Conference*.

Ian Soboroff. 2004. Overview of the TREC 2004 Novelty Track. In *TREC 2004 - the 13th Text Retrieval Conference*.

N. Stokes and J. Carthy. 2001. First story detection using a composite document representation. In *HLT 2001*, pages 134–141.

Wenyin Tang and Flora S Tsai. 2009. Threshold setting and performance monitoring for novel text mining. In *SIAM International Conference on Data Mining Workshop on Text Mining*.

Wenyin Tang, Agus Trisnajaya Kwee, and Flora S Tsai. 2009. Accessing contextual information for interactive novelty detection. In *European Conference on Information Retrieval (ECIR) Workshop on Contextual Information Access, Seeking and Retrieval Evaluation*.

Flora S. Tsai, Wenchou Han, Junwei Xu, and Hock Chuan Chua. 2009. Design and development of a mobile peer-to-peer social networking application. *Expert Syst. Appl.*, 36(8):11077 – 11087.

Mengqiu Wang, Kenji Sagae, and Teruko Mitamura. 2006. A fast, accurate deterministic parser for Chinese. In *ACL 2006, Sydney, Australia*, pages 425 – 432.

Youzheng Wu, Jun Zhao, and Bo Xu. 2003. Chinese named entity recognition combining a statistical model with human knowledge. In *ACL 2003 workshop on Multilingual and mixed-language named entity recognition*, pages 65–72.

Yiming Yang, Tom Pierce, and Jaime Carbonell. 1998. A study on retrospective and on-line event detection. pages 28–36. ACM Press.

Yiming Yang, Jian Zhang, Jaime Carbonell, and Chun Jin. 2002. Topic-conditioned novelty detection. In *SIGKDD 2002*, pages 688 – 693.

Huaping Zhang and Qun Liu. 2002. Model of Chinese words rough segmentation based on n-shortest paths method. *Journal of Chinese Information Processing*, 15:1–7.

Yi Zhang and Flora S. Tsai. 2009. Combining named entities and tags for novel sentence detection. In *ESAIR '09: Proceedings of the WSDM '09 Workshop on Exploiting Semantic Annotations in Information Retrieval*, pages 30–34.

Yi Zhang, Jamie Callan, and Thomas Minka. 2002. Novelty and redundancy detection in adaptive filtering. In *ACM SIGIR 2002, Tampere, Finland*, pages 81–88.

Le Zhao, Min Zheng, and Shaoping Ma. 2006. The nature of novelty detection. *Information Retrieval*, 9:527–541.

Wei Zheng, Yu Zhang, Bowei Zou, Yu Hong, and Ting Liu. 2008. Research of Chinese topic tracking based on relevance model.

Latent Document Re-Ranking

Dong Zhou[1,2] **Vincent Wade**[1]
1. University of Dublin, Trinity College, Dublin 2, Ireland
2. School of Computer and Communication, Hunan University, Changsha, Hunan, China
dongzhou1979@hotmail.com Vincent.Wade@cs.tcd.ie

Abstract

The problem of re-ranking initial retrieval results exploring the intrinsic structure of documents is widely researched in information retrieval (IR) and has attracted a considerable amount of time and study. However, one of the drawbacks is that those algorithms treat queries and documents separately. Furthermore, most of the approaches are predominantly built upon graph-based methods, which may ignore some hidden information among the retrieval set.

This paper proposes a novel document re-ranking method based on Latent Dirichlet Allocation (LDA) which exploits the implicit structure of the documents with respect to original queries. Rather than relying on graph-based techniques to identify the internal structure, the approach tries to find the latent structure of "topics" or "concepts" in the initial retrieval set. Then we compute the distance between queries and initial retrieval results based on latent semantic information deduced. Empirical results demonstrate that the method can comfortably achieve significant improvement over various baseline systems.

1 Introduction

Consider a traditional IR problem, where there exists a set of documents $\mathbb{D}$ in the collection. In response to an information need (as expressed in a query q), the system determines a best fit between the query and the documents and returns a list of retrieval results, sorted in a decreasing order of their relevancy. In practice, high precision at the top rankings of the returned results is of particular interest. Generally, there are two ways to automatically assist in achieving this ultimate goal after an initial retrieval process (Baeza-Yates and Ribeiro-Neto, 1999): document re-ranking and query expansion/re-weighting. Since the latter normally need a second round of retrieval process, our method focuses on the document re-ranking approach. We will focus on adjusting the ranking positions directly over initial retrieval results set $\mathbb{D}_{init}$.

Recently, there is a trend of exploring the hidden structure of documents to re-rank results. Some of the approaches represent the document entities as a connected graph G. It is usually constructed by links inferred from the content information as a nearest-neighbor graph. For example, Zhang et al. (2005) proposed an affinity ranking graph to re-rank search results by optimizing diversity and information richness. Kurland and Lee (2005) introduced a structural re-ranking approach by exploiting asymmetric relationships between documents induced by language models. Diaz (2005); Deng et al. (2009) use a family of semi-supervised machine learning methods among documents graph constructed by incorporating different evidences. However in this work we are more interested in adopting an automatic approach.

There are two important factors that should be taken into account when designing any re-ranking algorithms: the original queries and initial retrieval scores. One of issues is that previous structural re-ranking algorithms treat the query and the content individually when computing re-ranking scores. Each document is assigned a score independent of other documents without considering of queries. The problem we want to address in this paper is how we can leverage the interconnections between query and documents for the re-ranking purpose.

Another problem with such approaches concerns the fundamental re-ranking strategy they adopted. HITS (Kleinberg, 1999) and PageRank

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1571–1580, Singapore, 6-7 August 2009.

(Brin and Page, 1998) style algorithms were widely used in the past. However, approaches depend only on the structure of the global graph or sub-graph may ignore important information content of a document entity. As pointed out by Deng et al. (2009), re-ranking algorithms that rely only on the structure of the global graph are likely lead to the problem of topic drift.

Instead, we introduce a new document re-ranking method based on Latent Dirichlet Allocation (LDA) (Blei et al., 2003) which exploits implicit structure of the documents with respect to original queries. Rather than relying on graph-based techniques to identify the internal structure, the approach tries to directly model the latent structure of "topics" or "concepts" in the initial retrieval set. Then we can compute the distance between queries and initial retrieval results based on latent semantic information inferred. To prevent the problem of topic drift, the generative probability of a document is summed over all topics induced. By combining the initial retrieval scores calculated by language models, we are able to gather important information for re-ranking purposes. The intuition behind this method is the hidden structural information among the documents: *similar documents are likely to have the same hidden information with respect to a query*. In other words, if a group of documents are talking about the same topic which shares a strong similarity with a query, in our method they will get allocated similar ranking as they are more likely to be relevant to the query. In addition, the refined ranking scores should be relevant to the initial ranking scores, which, in our method, are combined together with the re-ranking score either using a linear fashion or multiplication process.

To illustrate the effectiveness of the proposed methodology, we apply the framework to ad-hoc document retrieval and compare it with the initial language model-based method and other three PageRank style re-ranking methods. Experimental results show that the improvement brought by our method is consistent and promising.

The rest of the paper is organized as follows. Related work on re-ranking algorithms and LDA based methods is briefly summarized in Section 2. Section 3 describes the re-ranking framework based on latent information induced together with details of how to build generative model. In Section 4 we report on a series of experiments performed over three different test collections in English and French as well as results obtained. Finally, Section 5 concludes the paper and speculates on future work.

2 Related Work

There exist several groups of related work in the areas of document retrieval and re-ranking.

The first category performs re-ranking by using inter-document relationship (Lee et al., 2001), evidences obtained from external resources (Kamps, 2004), or through local context analysis (Xu and Croft, 2000). In the past, document distances (Balinski and Daniowicz, 2005), manually built external thesaurus (Qu et al., 2001), and structural information (such as document title) (Luk and Wong, 2004), etc have been used extensively for this very purpose.

A second category of work is related to recent advances in structural re-ranking paradigm over graphs. Kurland and Lee performed re-ranking based on measures of centrality in the graph formed by generation links induced by language model scores, through a weighted version of PageRank algorithm (Kurland and Lee, 2005) and HITS-style cluster-based approach (Kurland and Lee, 2006). Zhang et al. (2005) proposed a similar method to improve web search based on a linear combination of results from text search and authority ranking. The graph, which they named affinity graph, shares strong similarities with Kurland and Lee's work with the links induced by a modified version of cosine similarity using the vector space model. Diaz (2005) used score regularization to adjust document retrieval rankings from an initial retrieval by a semi-supervised learning method. Deng et al. (2009) further developed this method. They built a latent space graph based on content and explicit links information. Unlike their approach we are trying to model the latent information directly.

This work is also related to a family of methods so called latent semantic analysis (LSA) (Landauer et al., 1998), especially topic models used for document representation. Latent Dirichlet Allocation (LDA), after it was first introduced by Blei et al. (2003), has quickly become one of the most popular probabilistic text modeling techniques and has inspired research ranging from text classification and clustering (Phan et al., 2008), information discovery (Mei et al., 2007; Titov and McDonald, 2008) to information retrieval (Wei and Croft, 2006). In this model, each topic is represented by a set of words and each word corresponds with a weight to measure its contribution to the topic. Wei and Croft

(2006) described large-scale information retrieval experiments by using LDA. In their work, LDA-based document model and language model-based document model were linearly combined to rank the entire corpus. However, unlike this approach we only apply LDA to a small set of documents. There are two reasons by doing so. One is the concern of computational cost. LDA is a very complex model and the complexity will grow linearly with the number of topics and the number of documents. Only running it through a document set significantly smaller than the whole corpus has obvious advantages. Secondly, it is well known that LSA-based method suffers from an incremental build problem. Normally adding new documents to the corpus needs to "be folded in" to the latent representation. Such incremental addition fails to capture the co-occurrences of the newly added documents (and even ignores all new terms they contain). As such, the quality of the LSA representation will degrade as more documents are added and will eventually require a re-computation of the LSA representation. Because our method only requires running LDA once for a small number of documents, this problems could be easily avoided. In addition, we also introduce two new measures to calculate the distance between a query and a document.

3 Latent Re-Ranking Framework

In this section, we describe a novel document re-ranking method based on extracting the latent structure among the initial retrieval set and measuring the distance between queries and documents.

3.1 Problem Definition

Let $\mathbb{D} = \{d_1, d_2, \dots, d_n\}$ denote the set of documents to be retrieved. Given a query q, a set of initial results $\mathbb{D}_{init} \in \mathbb{D}$ of top documents are returned by a standard information retrieval model (initial ranker). However, the initial ranker tends to be imperfect. The purpose of our re-ranking method is to re-order a set of documents $\mathbb{D}'_{init}$ so as to improve retrieval accuracy at the very top ranks of the final results.

3.2 Latent Dirichlet Allocation

We will first introduce Latent Dirichlet Allocation model which forms the basis of the re-ranking framework that will be detailed in the next subsection. It was previously shown that co-occurrence structure of terms in text documents can be used to recover some latent topic structures without any usage of background information (Landauer et al., 1998). This means that latent-topic representations of text allow modeling of linguistic phenomena such as synonymy and polysemy. By doing so, information retrieval systems can match the information needs with content items on a meaning level rather than by just lexical congruence.

The basic generative process of LDA closely resembles PLSA (Hofmann, 1999). LDA extends PLSA method by defining a complete generative model of text. The topic mixture is drawn from a conjugate Dirichlet prior that remains the same for all documents. The process of generating a document corpus is as follows:

1) Pick a multinomial distribution $\vec{\varphi}_z$ for each topic k from a Dirichlet distribution with hyperparameter $\vec{\beta}$.
2) For each document d, pick a multinomial distribution $\vec{\theta}_d$, from a Dirichlet distribution with hyperparameter $\vec{\alpha}$.
3) For each word token w in document d, pick a topic $z \in \{1 \dots k\}$ from the multinomial distribution $\vec{\theta}_d$.
4) Pick word w from the multinomial distribution $\vec{\varphi}_z$.

Thus, the likelihood of generating a corpus is:

$$p(d_1, \dots, d_n | \vec{\alpha}, \vec{\beta}) = \iint \prod_{d=1}^{n} p(\vec{\theta}_d | \vec{\alpha}) \cdot \prod_{z=1}^{k} p(\vec{\varphi}_z | \vec{\beta}) \cdot \prod_{i=1}^{N_d} \sum_{z_i=1}^{k} p(z_i | \vec{\theta}_d) p(w_i | z, \vec{\varphi}_z) \, d\vec{\theta}_d \, d\vec{\varphi}_z$$

Unlike PLSA model, LDA possesses fully consistent generative semantics by treating the topic mixture distribution as a k-parameter hidden random variable. LDA offers a new and interesting framework to model a set of documents. The documents and new text sequences (for example, queries) could be easily connected by "mapping" them to the topics in the corpus. In the next subsection we will introduce how to achieve this goal and apply it to document re-ranking.

LDA is a complex model and cannot be solved by exact inference. There are a few approximate inference techniques available in the literature: variational methods (Blei et al., 2003), expectation propagation (Griffiths and Steyvers, 2004)

and Gibbs sampling (Griffiths and Steyvers, 2004). Gibbs sampling is a special case of Markov-Chain Monte Carlo (MCMC) simulation and often yields relatively simple algorithms. For this reason, we choose to use Gibbs sampling to estimate LDA.

According to Gibbs sampling, we need to compute the conditional probability $p(z_i|\vec{z}_{\neg i}, \vec{w})$, where $\vec{w}$ denotes the vector of all words and $\vec{z}_{\neg i}$ denotes the vector of topic assignment except the considered word at position i. This probability distribution can be derived as:

$$p(z_i|\vec{z}_{\neg i}, \vec{w}) = \frac{n^{w_i}_{z,\neg i} + \beta_{w_i}}{(\sum_{v=1}^{V} n^{v}_{z} + \beta_v) - 1} \cdot \frac{n^{z}_{d_i,\neg i} + \alpha_k}{(\sum_{z=1}^{k} n^{z}_{d_i} + \alpha_z) - 1}$$

where $n^{t}_{z,\neg i}$ indicates the number of instances of word w_i assigned to topic $z = k$, not including the current token and $n^{z}_{d_i,\neg i}$ denotes the number of words in document d_i assigned to topic $z = k$, not including the current token.

Then we can obtain the multinomial parameter sets:

$$\theta_{d_i,k} = \frac{n^{k}_{d_i} + \alpha_k}{\sum_{z=1}^{k} n^{z}_{d_i} + \alpha_z}$$

$$\varphi_{k,w_i} = \frac{n^{w_i}_{k} + \beta_{w_i}}{\sum_{v=1}^{V} n^{v}_{z} + \beta_v}$$

The Gibbs sampling algorithm runs over three periods: initialization, burn-in and sampling. We do not tune to optimize these parameters because in our experiments the markov chain turns out to converge very quickly.

3.3 LDA-based Re-Ranking

Armed with this LDA methodology, we now describe the main idea of our re-ranking method. Given a set of initial results $\mathbb{D}_{init}$, we are trying to re-measure the distance between the query and a document. In the vector space model, this distance is normally the cosine or inner product measure between two vectors. Under the probabilistic model framework, this distance can be obtained from a non-commutative measure of the difference between two probability distributions. The distance used in our approach is the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951). Given two probability mass function $p(x)$ and $q(x)$, the KL divergence (or relative entropy) between p and q is defined as:

$$D(p||q) = \sum_{x} p(x) log \frac{p(x)}{q(x)}$$

In terms of text sequences (either queries or documents), the probability distribution can be regarded as a probabilistic language model M_d or M_q from each document d or each query q. In other words, it assumes that there is an underlying language model which "generates" a term (sequence) (Ponte and Croft, 1998). The unigram language model is utilized here. There are several ways to estimate the probabilities. Let $g(w \in d)$ denotes the number of times the term w occurs in a document d (same idea can be used on a query). The Maximum-likelihood estimation (MLE) of w with respect to d is defined as:

$$MLE_d w \stackrel{\text{def}}{=} \frac{g(w \in d)}{\sum_{w'} g(w' \in d)}$$

Previous work in language-model-based information retrieval (Zhai and Lafferty, 2004) advocates the use of a Dirichlet-smoothed estimation:

$$DIR_d w \stackrel{\text{def}}{=} \frac{g(w \in d) + \mu \cdot MLE_w \mathbb{D}}{\sum_{w'} g(w' \in d) + \mu}$$

where smoothing parameter μ controls the degree of reliance on relative frequencies in the document corpus rather than on the counts in d. The initial ranker that we choose to use later in the experiment computes the KL divergence between the $MLE_q w$ and a modified version of $DIR_d w$ (Zhai and Lafferty, 2001).

Both estimations can be easily extended to distributions over text sequences by assuming that the terms are independent:

$$MLE_d(w_1 w_2 \ldots w_n) \stackrel{\text{def}}{=} \prod_{j=1}^{n} MLE_d(w_j)$$

$$DIR_d(w_1 w_2 \ldots w_n) \stackrel{\text{def}}{=} \prod_{j=1}^{n} DIR_d(w_j)$$

In the re-ranking setting, we estimate that the probability of a document d generates w, using a mixture model LDA. It uses a convex combination of a set of component distributions to model observations. In this model, a word w is generated from a convex combination of some hidden topics z:

$$LDA_d(w) = \sum_{z=1}^{k} p(w|z) p(z|d)$$

where each mixture model $p(w|z)$ is a multinomial distribution over terms that correspond to one of the latent topics z. Similar to MLE and DIR estimations, this could be generated to give a distribution on a sequence of text:

Collection	Contents	Language	Num of docs	Size	Queries
BL (CLEF2008)	British Library Data	English (Main)	1,000,100	1.2 GB	50
BNF (CLEF2008)	Bibliothèque Nationale de France	French (Main)	1,000,100	1.3 GB	50
LAT (CLEF2007)	Los Angeles Times 2002	English	135,153	434 MB	50

Table 1. Statistics of test collections

$$LDA_d(w_1 w_2 \dots w_n) \stackrel{\text{def}}{=} \prod_{j=1}^{n} LDA_d(w_j)$$

Then the distance between a query and a document based on this model can be obtained. The first method we propose here adopts the KL divergence between the query terms and document terms to compute a Re-Rank score RS_{LDA}^{KL1}:

$$RS_{LDA}^{KL1} = -D(MLE_q(\cdot)||LDA_d(\cdot))$$

This method also has the property of length-normalization to ameliorate long document bias problems (Kurland and Lee, 2005).

The second method also measures a KL divergence between a query and a document, however, in a different way. As in the original LDA model, the multinomial parameter $\vec{\theta}_d$ indicates the topic distribution of a document d. Query q can be considered as topic estimation of a unknown document $\tilde{\vec{w}}$. Thus by first randomly assigning topics to words and then performing a number of loops through the Gibbs sampling update, we have:

$$p(z_i|\tilde{\vec{z}}_{\neg i}, \tilde{\vec{w}}; \vec{z}_{\neg i}, \vec{w}) = \frac{n_{z,\neg i}^{w_i} + \tilde{n}_{z,\neg i}^{w_i} + \beta_{w_i}}{(\sum_{v=1}^{V} n_z^v + \tilde{n}_z^v + \beta_v) - 1} \cdot \frac{n_{\tilde{d}_i,\neg i}^{z} + \alpha_k}{(\sum_{z=1}^{k} n_{\tilde{d}_i}^z + \alpha_z) - 1}$$

where $\tilde{n}_{z,\neg i}^{w_i}$ counts the observations of word w_i and topic k in unseen document. Then the topic distribution for the query (just the unseen document $\tilde{d}_i$) is:

$$\tilde{\theta}_{\tilde{d}_i,k} = \frac{n_{\tilde{d}_i}^k + \alpha_k}{\sum_{z=1}^{k} n_{\tilde{d}_i}^z + \alpha_z}$$

so that the distance between a query q and a document d is defined as the KL divergence between the topic distributions of q and d. Then the re-ranking score is calculated as:

$$RS_{LDA}^{KL2} = -D(\vec{\theta}_q||\vec{\theta}_d)$$

Thus we can re-rank the initial retrieved documents according to the scores acquired. However, as in other topic models, a topic in the LDA model represents a combination of words, and it may not be as precise a representation as words in language model. Hence we need to further consider how to combine initial retrieval scores with the re-ranking scores calculated. Two combination methods will be presented in the next subsection.

3.4 Combining Initial Retrieval Scores

Motivated by the significant improvement obtained by (Wei and Croft, 2006) and (Zhang et al., 2005), we formulate our method through a linear combination of the re-ranking scores based on initial ranker and the latent document re-ranker, shown as follow:

$$RS1 = (1-\lambda) \cdot OS + \lambda \cdot RS_{LDA}^{KL};$$

where OS denotes original scores returned by the initial ranker and λ is a parameter that can be tuned with $\lambda = 0$ meaning no re-ranking is performed.

Another scheme considers a multiplication combination to incorporate the original score. It does not need to tune any parameters:

$$RS2 = OS \cdot RS_{LDA}^{KL};$$

This concludes our overview of the proposed latent re-ranking method.

4 Evaluation

In this section, we will empirically study the effectiveness of the latent document re-ranking method over three different data collections.

4.1 Experimental Setup

Data The text corpus used in our experiment was made up from elements of the CLEF-2007 and CLEF-2008 the European Library (TEL) collections[1] written in English and French. These collections are described in greater detail in Table 1. All of the documents in the experiment were indexed using the Lemur toolkit[2]. Prior to

[1] http://www.clef-campaign.org
[2] http://www.lemurproject.org

indexing, Porter's stemmer and a stopword list[3] were used for the English documents. We use a French analyzer[4] to analyze French documents.

It is worth noting that the CLEF-2008 TEL data is actually multilingual: all collections to a greater or lesser extent contain records pointing to documents in other languages. However this is not a major problem because the majority of documents in the test collection are written in main languages of those test collections (BL-English, BNF-French). Furthermore, documents written in different languages tend not to match the queries in main languages. Also the data is very different from the newspaper articles and news agency dispatches previously used in the CLEF as well as TREC[5]. The data tends to be very sparse. Many records contain only title, author and subject heading information; other records provide more detail. The average document lengths are 14.66 for BL and 24.19 for BNF collections after pre-processing, respectively. Please refer to (Agirre et al., 2008) for a more detailed discussion about this data. The reason we choose these data collections is that we wanted to test the scalability of the proposed method in different settings and over different guages. In addition we also select a more tional collection (LAT from CLEF2007) as a test base.

We also used the CLEF-2007 and CLEF-2008 query sets. The query sets consist of 50 topics in English for LAT, BL and in French for BNF, all of which were used in the experiment. Each topic is composed of several parts such as: *Title*, *Description*, *Narrative*. We chose to conduct *Title+Description* runs as queries. The queries are processed similarly to the treatment in the test collections. The relevance judgments are taken from the judged pool of top retrieved documents by various participating retrieval systems from previous CLEF workshops.

We compare the proposed latent re-ranking method with four other approaches: the initial ranker, mentioned above, is a KL-divergence retrieval function using the language models. Three other baseline systems are: Kurland and Lee's structural re-ranking approach (Recursive Weighted Influx + Language Model), chosen as it demonstrates the best performance in their paper (Kurland and Lee, 2005), Zhang et al.'s affinity graph-based approach (Zhang et al., 2005) and a variant of Kurland and Lee's work with links in the graph calculated by the vector-space model (cosine similarity as mentioned in (Kurland and Lee, 2005)). We denote these four systems as InR, RWILM, AFF, and VEC respectively. Furthermore, we denote the permutations of our methods as follows: LDA1: $RS2\ with\ RS_{LDA}^{KL1}$, LDA2: $RS1\ with\ RS_{LDA}^{KL1}$, LDA3: $RS2\ with\ RS_{LDA}^{KL2}$, LDA4: $RS1\ with\ RS_{LDA}^{KL2}$.

Because the inconsistency of the evaluation metrics employed in the past work, we choose to employ all of them to measure the effectiveness of various approaches. These include: mean average precision (MAP), the precision of the top 5 documents (Prec@5), the precision of the top 10 documents (Prec@10), normalized discounted cumulative gain (NDCG) (Jarvelin and Kekalainen, 2002) and Bpref (Buckley and Voorhees, 2004). Statistical-significant differences in performance were determined using a paired t-test at a confidence level of 95%.

It is worth pointing out that the above measurements are not directly comparable with those of the CLEF participants because we restricted our initial pool to a smaller number of documents and the main purpose in the paper is to compare the proposed method with different baseline systems.

Parameter Two primary parameters need to be determined in our experiments. For the re-ranking experiments, the combination parameter λ must be defined. For the LDA estimation, the number of topics k must be specified. We optimized settings for these parameters with respect to MAP, not with all other metrics over the BL collection and apply them to all three collections directly.

The search ranges for these two parameters were:

λ : 0.1, 0.2, …, 0.9
k : 5, 10, 15, …, 45

As it turned out, for many instances, the optimal value of λ with respect to MAP was either 0.1 or 0.2, suggesting the initial retrieval scores have valuable information inside them. In contrast, the optimal value of k was between 20 and 40. Although this demonstrates a relatively large variance, the differences in terms of MAP have remained small and statistically insignificant. We set $\mathbb{D}_{init}$ to 50 in all results reported, as in Kurland and Lee's paper (Kurland and Lee, 2005) and we later show that the performance turns out to be very stable when this set enlarged.

[3] ftp://ftp.cs.cornell.edu/pub/smart/
[4] http://lucene.apache.org/
[5] http://trec.nist.gov/

	BL				
	MAP	Prec@5	Prec@10	NDCG	Bpref
InR	0.1913	0.52	0.452	0.3489	0.2287
RWILM	**0.2152**	0.532	0.468	0.3663	0.2242
AFF	0.1737	0.444	0.434	0.3273	0.22
VEC	0.1756	0.448	0.434	0.3258	0.2216
LDA1	0.21 o, a, v	*0.544 a, v*	*0.47*	*0.3679 o, a, v*	*0.2429 a, v*
LDA2	0.2148 o, a, v	***0.58 o, a, v***	***0.5 o, a, v***	***0.3726 o, a, v***	***0.2491 o, l, a, v***
LDA3	0.1673	0.452	0.402	0.3297	0.2
LDA4	0.2035 o, a, v	*0.548 a, v*	0.468 a, v	0.3626 o, a, v	*0.2326 a*
	BNF				
	MAP	Prec@5	Prec@10	NDCG	bpref
InR	0.1266	0.268	0.216	0.2456	0.1482
RWILM	0.1274	0.264	0.218	0.2495	0.1498
AFF	0.108	0.248	0.21	0.2221	0.1404
VEC	0.1126	0.252	0.214	0.2262	0.1463
LDA1	*0.1374 a, v*	*0.292 a*	*0.242*	*0.2544 a, v*	*0.1617*
LDA2	***0.1452 o, a, v***	***0.292 a, v***	*0.244 a*	***0.2608 o, a, v***	***0.1697 o, l, a, v***
LDA3	0.1062	0.232	0.202	0.2226	0.1439
LDA4	*0.1377 a,v*	*0.28 a*	***0.246 o, a, v***	*0.2507 a, v*	*0.1672 o, a, v*
	LAT02				
	MAP	Prec@5	Prec@10	NDCG	bpref
InR	0.3119	0.568	0.48	0.5093	0.3105
RWILM	0.3097	0.556	0.478	0.5096	0.3064
AFF	0.3065	0.572	0.492	0.5037	0.312
VEC	0.301	0.536	0.474	0.4975	0.3087
LDA1	*0.3253 v*	***0.584 v***	***0.502 v***	*0.5158 v*	*0.3339 o, l, v*
LDA2	*0.3271 a, v*	***0.584 o, v***	*0.496*	*0.518 o, v*	*0.3351 o, l, a, v*
LDA3	0.2848	0.444	0.398	0.486	0.2879
LDA4	***0.3274 o***	0.552	0.478	***0.5202 o, v***	***0.3396 o, l, v***

Table 2. Experimental Results. For each evaluation setting, improvements over the RWILM baseline are given in italics (because it has highest performance); statistically significant differences between our methods and InR, RWILM, AFF, VEC are indicated by o, l, a, v, respectively. Bold highlights the best results over all algorithms.

Lastly, the parameters in the baseline systems are set according to the tuning procedures in their original papers[6].

[6] More specifically, the combination parameter was set to 0.5 for AFF, the number of links was set to 4 for RWILM.

4.2 Results

Primary Evaluation The main experimental results are presented in Table 2. The first four rows in each collection specify reference-comparison data. The first question we are interested in is how our latent re-ranking methods

perform (taken as a whole). It is shown that our methods bring improvements upon the various baselines in 75% of the 48 relevant comparisons (4 latent re-ranking methods × 4 corpora × 4 baselines). Only the algorithm permutation LDA3 performs less well. Furthermore, our methods are able to achieve the highest performance across all the evaluation metrics over three test collections except in one case (MAP in BL collection). An even more exciting observation is that in many cases, our methods, even though tuned for MAP, can outperform various baselines for all the evaluation metrics, with statistically significant improvements in many runs.

A closer examination of the results in Table 2 reveals some interesting properties. As expected, the RWILM method bought improvements in many cases in CLEF-2008 test collections. However, the performance over CLEF-2007 collection was somewhat disappointing. This seems to indicate that the language model induced graph method tends to perform better in sparse data rather than longer documents. Also Language Modeling requires large set training data to be effective, while the complexity of our method is only linear with number of topics and the number of documents for each iteration. The affinity and vector graph based methods demonstrated poor performance across all the collections. This may be due to the fact that the approach Zhang et al. (Zhang et al., 2005) developed focuses more on diversity and information richness and cares less about the precision of the retrieval results while asymmetric graph as constructed by the vector space model fails in capturing important relationship between the documents.

Another observation we can draw from Table 2 is that the relative performance tends to be stable during test collections written in different languages. This shows a promising future for studying structure of the documents with respect to queries for re-ranking purpose. At the same time, efficiency is always an issue in all re-ranking methods. Although this is not a primary concern in the current work, it would definitely worth thinking in the future.

We also conducted some experiments over queries constructed by using *Title* field only. This forms some more realistic short queries. The experiments showed very similar results compared to longer queries. This demonstrates that the query length is a trivial issue in our methods (as in other graph-based structural re-ranking). We examined the best and worse performed queries, their performance are generally consistent across all the methods. This phenomenon should be investigated further in the follow up evaluation.

Comparison of Different Methods In comparison of performance between four permutations of our methods, LDA2 is the clear winner over CLEF-2008 test collections. The results obtained by LDA2 and LDA4 over CLEF-2007 test collection were mixed. LDA2 performed better in precision at top n documents while LDA4 showed promising results in terms of more general evaluation metrics. On the other hand, the linear combination approach performed much better than multiplication based combination. The situation is even worse when we adopted the RS_{LDA}^{KL2} method, which was inferior in several cases. Thus the linear combination should be highly recommended.

Scalability We have shown that our latent document re-ranking method is successful at accomplishing the goal of improving the results returned by an initial retrieval engine. But one may raise a question of whether it is necessary to restrict our attention to an initial pool $\mathbb{D}_{init}$ at such a small size. As it happens, preliminary experiments with LDA2 on larger size of the initial pool are presented in Figure 1. As we can see, our method can bring consistently stable improvements.

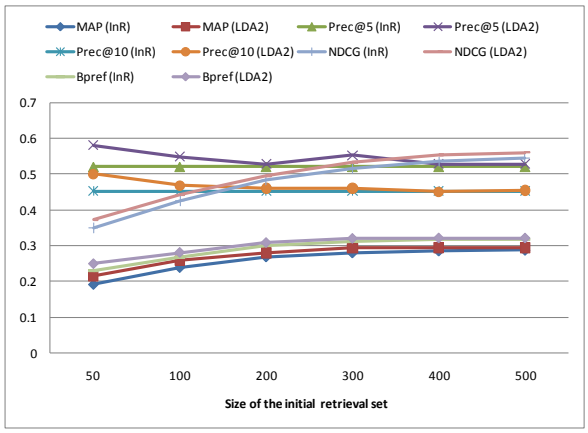


Figure 1. Experiments with larger initial pools

5 Conclusion and Future Work

In this paper we proposed and evaluated a latent document re-ranking method for re-ordering the initial retrieval results. The key to refine the results is finding the latent structure of "topics" or "concepts" in the document set, which leve-

rages the latent Dirichlet allocation technique for the query-dependent ranking problem and results in state-of-art performance.

There are many research directions we are planning to investigate. It has been shown that LDA-based retrieval is a promising method for ranking the whole corpus. There is a desire to call for a direct comparison between ranking and re-ranking using the proposed algorithmic variations. Future work will also include the comparison between our methods with other related approaches, such as Kurland and Lee's cluster-based approach (Kurland and Lee, 2006).

There exist a sufficient number of latent semantic techniques such as singular vector decomposition, non-negative matrix factorization, PLSA, etc. We are planning to explore these methods to compare their performance. Also direct re-ranking can be used to improve automatic query expansion since better ranking in top retrieved documents can be expected to improve the quality of the augmented query. We believe this is another fruitful line for future research.

Acknowledgments

The authors would like to thank three anonymous reviewers for many constructive comments. This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at University of Dublin, Trinity College**.**

References

Eneko Agirre, Giorgio M. Di Nunzio, Nicola Ferro, Thomas Mandl and Carol Peters (2008). CLEF 2008: Ad Hoc Track Overview. In *Working notes of CLEF2008*.

Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto (1999). *Modern Information Retrieval*, Addison-Wesley Longman Publishing Co., Inc.

Jaroslaw Balinski and Czeslaw Daniowicz (2005). "Re-ranking method based on inter-document distances." *Inf. Process. Manage.* **41**(4): 759-775.

David M. Blei, Andrew Y. Ng and Michael I. Jordan (2003). "Latent dirichlet allocation." *J. Mach. Learn. Res.* **3**: 993-1022.

Sergey Brin and Lawrence Page (1998). "The anatomy of a large-scale hypertextual Web search engine." *Comput. Netw. ISDN Syst.* **30**(1-7): 107-117.

Chris Buckley and Ellen M. Voorhees (2004). Retrieval evaluation with incomplete information. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, Sheffield, United Kingdom, ACM. p. 25-32.

Hongbo Deng, Michael R. Lyu and Irwin King (2009). Effective latent space graph-based re-ranking model with global consistency. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, Barcelona, Spain, ACM. p. 212-221.

Fernando Diaz (2005). Regularizing ad hoc retrieval scores. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, Bremen, Germany, ACM. p. 672-679.

Thomas L. Griffiths and Mark Steyvers (2004). Finding scientific topics. In *Proceeding of the National Academy of Sciences*. p. 5228-5235.

Thomas Hofmann (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, Berkeley, California, United States, ACM. p. 50-57.

Kalervo Jarvelin and Jaana Kekalainen (2002). "Cumulated gain-based evaluation of IR techniques." *ACM Trans. Inf. Syst.* **20**(4): 422-446.

Jaap Kamps (2004). Improving Retrieval Effectiveness by Reranking Documents Based on Controlled Vocabulary In *Proceedings of 26th European Conference on IR Research, ECIR 2004*, Sunderland, UK. p. 283-295.

Jon M. Kleinberg (1999). "Authoritative sources in a hyperlinked environment." *J. ACM* **46**(5): 604-632.

S. Kullback and R. A. Leibler (1951). "On Information and Sufficiency." *The Annals of Mathematical Statistics* **22**(1): 79-86.

Oren Kurland and Lillian Lee (2005). PageRank without hyperlinks: structural re-ranking using links induced by language models. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, Salvador, Brazil, ACM. p. 306-313.

Oren Kurland and Lillian Lee (2006). Respect my authority!: HITS without hyperlinks, utilizing cluster-based language models. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, Seattle, Washington, USA, ACM. p. 83-90.

Thomas K. Landauer, Peter W. Foltz and Darrell Laham (1998). "An Introduction to Latent Semantic Analysis." *Discourse Processes* **25**: 259-284.

Kyung-Soon Lee, Young-Chan Park and Key-Sun Choi (2001). "Re-ranking model based on document clusters." *Inf. Process. Manage.* **37**(1): 1-14.

Robert W. P. Luk and K.F. Wong (2004). Pseudo-Relevance Feedback and Title Re-ranking for Chinese Information Retrieval. In *Working Notes of the Fourth NTCIR Workshop Meeting*, Tokyo, Japan, National Institute of Informatics.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su and ChengXiang Zhai (2007). Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web*, Banff, Alberta, Canada, ACM. p. 171-180.

Xuan-Hieu Phan, Le-Minh Nguyen and Susumu Horiguchi (2008). Learning to classify short and sparse text \& web with hidden topics from large-scale data collections. In *Proceeding of the 17th international conference on World Wide Web*, Beijing, China, ACM. p. 91-100.

Jay M. Ponte and W. Bruce Croft (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, Melbourne, Australia, ACM. p. 275-281.

Youli Qu, Guowei Xu and Jun Wang (2001). Rerank Method Based on Individual Thesaurus. In *Proceedings of the Second NTCIR Workshop on Research in Chinese & Japanese Text Retrieval and Text Summarization*, Tokyo, Japan, National Institute of Informatics.

Ivan Titov and Ryan McDonald (2008). Modeling online reviews with multi-grain topic models. In *Proceeding of the 17th international conference on World Wide Web*, Beijing, China, ACM. p. 111-120.

Xing Wei and W. Bruce Croft (2006). LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, Seattle, Washington, USA, ACM. p. 178-185.

Jinxi Xu and W. Bruce Croft (2000). "Improving the effectiveness of information retrieval with local context analysis." *ACM Trans. Inf. Syst.* **18**(1): 79-112.

Chengxiang Zhai and John Lafferty (2001). Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, Atlanta, Georgia, USA, ACM. p. 403-410.

Chengxiang Zhai and John Lafferty (2004). "A study of smoothing methods for language models applied to information retrieval." *ACM Trans. Inf. Syst.* **22**(2): 179-214.

Benyu Zhang, Hua Li, Yi Liu, Lei Ji, Wensi Xi, Weiguo Fan, Zheng Chen and Wei-Ying Ma (2005). Improving web search results using affinity graph. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, Salvador, Brazil, ACM. p. 504-511.

Author Index